Calcolatori Elettronici - Architettura e Organizzazione

Appendice A

Sistemi digitali

Giacomo Bucci

Revisione del 31 marzo 2017

Questo documento è una appendice al volume
Calcolatori Elettronici - Architettura e Organizzazione
IV edizione
McGraw-Hill Education (Italy), S.r.l.
Milano, 2017

Storia degli aggiornamenti

Marzo 2017: primo rilascio.

A

Sistemi digitali

OBIETTIVI

- Introdurre i circuiti logici e l'algebra delle reti logiche
- Illustrare le reti combinatorie e le reti sequenziali
- Presentare alcuni componenti di uso ricorrente
- Fornire cenni sulle tecnologie digitali

CONCETTI CHIAVE

Segnali binari, algebra di Boole, reti combinatore e reti sequenziali, circuiti logici, porte logiche, elementi di memoria binari (Flip-Flop), decodificatori, codificatori, selettori, memorie ROM, logica programmabile, registri, trasferimenti tra registri, bus.

INTRODUZIONE

In estrema sintesi, un calcolatore è un gigantesco sistema digitale. Per sistema digitale si intende una macchina che effettua processi di elaborazione su *segnali digitali*, ovvero su segnali che possono solo assumere valori da un insieme finito. I sistemi di nostro interesse sono quelli in cui l'insieme di definizione è limitato a due soli valori, cioè quei sistemi in cui i segnali (e le variabili che li rappresentano) assumono valori su di un insieme *binario*. Le reti corrispondenti sono descritte attraverso un'algebra, detta algebra delle reti, ovvero algebra di commutazione. Essa costituisce una interpretazione dell'algebra booleana, così denominata dal nome del matematico e filosofo inglese George Boole che la introdusse con un trattato del 1854, intitolato "*An Investigation into the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*". Proprio in virtù dell'origine di questa algebra, le reti in questione vengono usualmente dette "reti logiche".

L'impiego dell'algebra di Boole per descrivere le reti di commutazione venne proposto da Claude Shannon nell'articolo intitolato "*A Symbolic Analysis of Relay and Switching Circuits*", derivante dalla sua tesi di laurea presso il MIT (*Massachusetts Institute of Technology*), pubblicato nel 1938; qualche anno dopo Shannon sarebbe passato ai Bell Labs e avrebbe dato un contributo fondamentale alla teoria dell'informazione.

Questa appendice è dedicata all'illustrazione della logica dei sistemi digitali. Viene introdotta l'algebra di commutazione e vengono esposte le sue principali proprietà. Vengono discusse le reti combinatorie e le reti sequenziali. Vengono mostrati alcuni moduli di uso ricorrente. Per ulteriori approfondimenti si può consultare uno dei tanti testi sull'argomento [Bar91], [FSS07], [KJ10], [MK08], [Won85].

A.1 Logica dei sistemi digitali

Un *segnale analogico* porta più informazione di un segnale digitale, sia perché ad esso corrisponde una infinità di valori nell'intervallo di definizione, sia perché sono possibili una infinità di andamenti della forma d'onda. Nella pratica, disturbi e limiti fisici alla velocità di variazione rendono inaccurato sia il processo di generazione sia quello di riconoscimento e misura dei segnali analogici. Al contrario, i *segnali digitali* sono molto meno sensibili al rumore e ai fenomeni transitori. Per quanto riguarda i disturbi, basta che questi non portino il segnale al di fuori della fascia di discretizzazione in cui si trova il valore del segnale al momento del verificarsi del disturbo. Per quanto riguarda i fenomeni transitori, basta dar tempo sufficiente al loro esaurirsi in modo che il riconoscimento del segnale avvenga in modo non ambiguo.

Un semplice ragionamento spiega bene le ragioni del successo dei sistemi digitali. Se un sistema si compone di n elementi e p è la probabilità che ha ciascuno di essi di operare correttamente, allora la probabilità che l'insieme degli n componenti operi correttamente è p^n. Se si considera che il numero di componenti elementari che formano un moderno calcolatore elettronico supera facilmente il miliardo, è necessario che p sia tendenzialmente 1. Da un punto di vista tecnologico ciò richiede componenti estremamente semplici e tali per cui sia facile diagnosticare le situazioni di malfunzionamento. Il massimo della semplicità e della affidabilità si ha con *sistemi digitali binari*, cioè con sistemi in cui i segnali possono assumere solo due valori e ai cui componenti è richiesto solo la capacità di discriminarli. Si tratta, per esempio, di riconoscere se un dato segnale è ad un livello di tensione alto o basso, ovvero se ha un valore superiore o inferiore ad una data soglia.

In Figura A.1, nell'angolo in alto a sinistra, viene riportato un possibile andamento di un segnale binario reale. Si fa riferimento a un segnale di tensione. Il segnale può stare

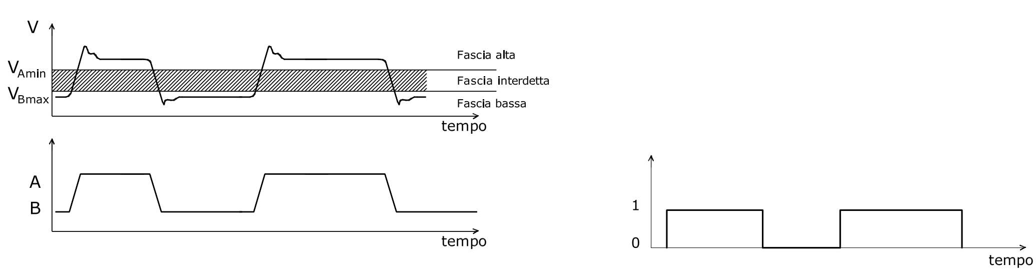


Figura A.1 A sinistra: possibile andamento di un segnale binario reale e sua schematizzazione. A destra: segnale binario teorico.

in due zone distinte, corrispondenti alla fascia alta e alla fascia bassa, delimitate, rispettivamente, da V_{Amin} e V_{Bmax}. La zona interdetta viene solo attraversata nel passaggio tra le due fasce. Nell'angolo in basso a sinistra viene mostrata la schematizzazione del segnale, come normalmente è dato di trovare nei documenti di specifica degli apparati digitali; questa schematizzazione vuole evidenziare il fatto che i tempi di *salita* e di *discesa* non sono nulli. Infine, in basso a destra viene riportato l'andamento di un teorico segnale binario. I due simboli 0 e 1 rappresentano i due valori "logici" che il segnale può assumere[1].

[1] Di norma si associa il simbolo 1 al livello di tensione alto e 0 al livello di tensione basso e si parla

In Figura A.2 viene schematizzata una rete logica. Essa trasforma l'insieme $I = \{x_1, x_2, \ldots \ldots, x_n\}$ dei segnali (binari) di ingresso nell'insieme $O = \{z_1, z_2, \ldots, z_m\}$ dei segnali (binari) di uscita. Una rete ha sempre comportamento unidirezionale, ovvero non è possibile alterare in alcun modo l'informazione presente ai *morsetti* di ingresso agendo sui morsetti di uscita.

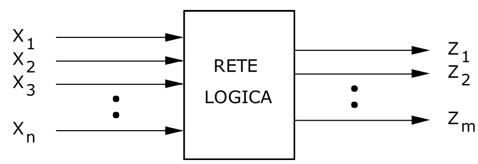


Figura A.2 Schematizzazione di una rete logica.

Si distinguono due categorie fondamentali di reti: le reti combinatorie e le reti sequenziali.

Reti combinatorie. Si ha una rete combinatoria quando l'uscita è esclusivamente funzione dell'ingresso, ovvero quando per una data configurazione dei segnali di ingresso risultano definiti i segnali di uscita. Formalmente: $O = f(I)$.

Reti sequenziali. Si ha una rete sequenziale quando l'uscita è funzione, oltre che dell'ingresso, anche dello *stato*. Indicando con $S = \{y_1, y_2, \ldots, y_l\}$ lo stato corrente della rete, l'uscita viene espressa come $O = f(I, S)$. Alla funzione di uscita è necessario aggiungere la funzione di stato, espressa come $S_f = g(I, S)$, che lega lo stato futuro allo stato presente e all'ingresso. Una rete sequenziale ha la cratteristica di avere *memoria* (limitata) della sequenza di ingresso.

I componenti elementari delle reti vengono detti "circuiti logici" o, più comunemente "porte". La Figura A.3 mostra i simboli delle tre porte logiche fondamentali. Le reti costruite a partire dalle porte logiche vengono descritte attraverso l'algebra booleana.

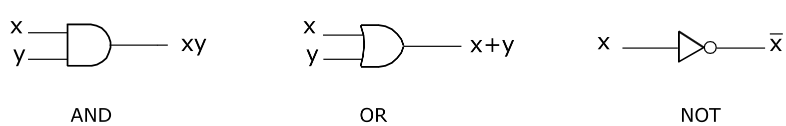


Figura A.3 Simboli standard per le porte AND, OR e NOT.

A.1.1 Algebra delle reti

L'algebra è definita su un insieme, composto da due soli elementi, convenzionalmente denotati con 0 e 1. I due simboli nulla hanno a che fare con i corrispondenti simboli aritmetici. Ciò che serve per definire l'algebra sono due simboli distinti. In altri contesti

di "logica positiva"; se invece viene associato 1 al livello di tensione basso e 0 a quello alto si parla di "logica negativa".

vengono usate differenti coppie di simboli come: vero/falso, v/f, T/F e altri. Come vedremo, l'uso dei simboli 0 e 1 risulta conveniente quando si tratta di impiegare le reti per la realizzazione degli operatori dell'aritmetica binaria, facendo corrispondere i simboli logici 0 e 1 ai numeri 0 e 1, in modo da poter realizzare l'aritmetica binaria attraverso reti logiche.

Sull'insieme {0,1} sono definite tre operazioni.

- Il prodotto logico, indicato con il segno "·". Il prodotto logico corrisponde al connettivo "e" (AND) in quanto afferma che *il risultato è 1 se e solo se sono 1 ambedue i termini del prodotto.*
- La somma logica, indicata con il segno "+". La somma logica corrisponde al connettivo "o" (OR) in quanto afferma che *il risultato è 1 se è 1 almeno uno dei due termini della somma.*
- la complementazione o negazione, indicata con il segno "$^-$". La complementazione corrisponde al connettivo "non" (NOT) in quanto afferma che *il risultato è 1 se il termine a cui si applica è 0 ed è 0 se il termine a cui si applica è 1.*

Anche per gli operatori · e + valgono le stesse avvertenze date per i due simboli 0 e 1: niente hanno a che vedere con i corrispondenti operatori aritmetici. I simboli "·", "+" e "$^-$" sono intercambiabili con AND, OR e NOT.

In Figura A.4 vengono riportate le tabelle di verità delle tre operazioni. La *tabella di verità* di una operazione riporta il risultato dell'operazione stessa per tutte le possibili combinazioni dei valori degli operandi.

Prodotto logico		
A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

Somma logica		
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Negazione	
A	$\overline{B}$
0	1
1	0

Figura A.4 Tabelle di verità delle tre operazioni fondamentali dell'algebra.

Si definisce *costante logica* (o *booleana*) un simbolo cui è permanentemente assegnato uno dei due possibili valori presi dall'insieme {0, 1}.

Si definisce *variabile logica* un simbolo che può assumere indifferentemente uno dei due valori presi dall'insieme {0, 1}. Se x è una variabile logica si ha:

$$x = 0 \quad \text{se e solo se} \quad x \neq 1 \qquad\qquad x = 1 \quad \text{se e solo se} \quad x \neq 0$$

Si definisce *espressione logica* una qualunque combinazione di variabili o costanti booleane legate fra loro dagli operatori logici fondamentali. I seguenti sono esempi di espressioni logiche.

$$x + x \cdot y \qquad x_1 \cdot x_2 + 1 \cdot x_3 \qquad x + \overline{y} \cdot 0 \qquad x \cdot (\overline{y} + w \cdot (z + 0)) + \overline{v}$$

Si definisce *funzione logica* delle n variabili booleane $x_1, \ldots, x_n$, la relazione $\{0, 1\}^n \rightarrow \{0, 1\}$ che associa un valore booleano a ciascuna delle 2^n configurazioni possibili delle n variabili:

$$y = f(x_1, ..., x_n)$$

Una funzione può essere espressa in forma algebrica o in forma tabellare. Ad esempio, la tabella e la forma algebrica di Figura A.5 sono equivalenti.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$$y = f(x_1, x_2, x_3) = \overline{x_1} \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3$$

Figura A.5 Esempio di tabella di verità e forma algebrica di una funzione booleana di tre varabili. Sulle tabelle di verità le configurazioni delle variabili sono riportate ordinatamente per riga. Nel caso specifico, si hanno 8 righe, numerate da 0 a $2^3 - 1 = 7$. Il modo in cui è stata scritta la forma algebrica a partire dalla tabella di verità sarà illustrato al Paragrafo A.2.

A.1.2 Proprietà dell'algebra

Vengono ora date alcune importanti proprietà dell'algebra[2].

1. Proprietà di idempotenza:

$$x + x = x$$
$$x \cdot x = x$$

   Prova (per induzione perfetta):

$$1 + 1 = 1 \qquad 0 + 0 = 0 \qquad 1 \cdot 1 = 1 \qquad 0 \cdot 0 = 0$$

2. Proprietà distributiva:

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$
$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

   Questa proprietà afferma che non solo il prodotto è distributivo rispetto alla somma, ma che anche la somma è distributiva rispetto al prodotto. Assieme alla precedente, questa proprietà marca una netta differenziazione rispetto all'aritmetica.

[2]Quando è possibile, il modo sicuro per provare le proprietà dell'algebra consiste nel ricorrere al metodo dell'*induzione perfetta*, ovvero nell'effettuare tutte le possibili sostituzioni di 0 e 1 per le variabili che compaiono nelle espressioni.

3. Proprietà associativa:

$$x + (y + z) = (x + y) + z$$
$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

4. Proprietà commutativa:

$$x + y = y + x$$
$$x \cdot y = y \cdot x$$

5. Proprietà di assorbimento:

$$x + x \cdot y = x$$
$$(x + y) \cdot x = x$$

Dimostrazione: $x + x \cdot y = x \cdot 1 + x \cdot y = x \cdot (1 + y) = x \cdot 1 = x$

6. Se x è una variabile booleana sono valide:

$$0 + x = x$$
$$1 \cdot x = x$$

7. Valgono le seguenti relazioni:

$$x\overline{x} = 0$$
$$x + \overline{x} = 1$$

8. Doppia negazione:

$$\overline{\overline{x}} = x$$

9. Elementi forzanti delle due operazioni AND e OR:

$$x + 1 = 1$$
$$x \cdot 0 = 0$$

10. Teorema di De Morgan:

$$\overline{x + y} = \overline{x} \cdot \overline{y}$$
$$\overline{x \cdot y} = \overline{x} + \overline{y}$$

Il teorema di De Morgan per due variabili si dimostra per induzione perfetta e vale anche nel caso generale di n variabili.

Si osservi che tutti i postulati e tutte le proprietà fin qui enunciati sono stati dati in coppia. Per ciascuna coppia si passa dall'una all'altra regola se si scambiano tra loro le operazioni di somma e prodotto e gli 1 con gli 0. È questo il *principio di dualità*. Esso ci permette di affermare che trovata una regola esiste la duale[3]

È già stato detto che ai tre operatori dell'algebra corrispondono le tre porte fondamentali (AND, OR, NOT). Dunque ogni espressione algebrica può essere trasformata in uno schema e viceversa. Il passaggio dalla rappresentazione algebrica a quella schematica è banale: basta sostituire gli operatori algebrici con le corrispondenti porte. Altrettanto ovvio risulta il passaggio dalla rappresentazione schematica a quella algebrica. In Figura A.6 viene mostrato un esempio di corrispondenza tra reti e forme algebriche.

[3] La proprietà 8 è *autoduale*.

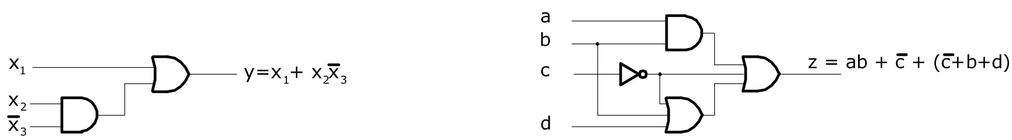


Figura A.6 Corrispondenza tra reti ed espressioni algebriche.

A.2 Forme canoniche e minimizzazione

Sia data una funzione in forma tabellare e ci si ponga il problema di trovare per essa un'espressione algebrica. Partendo dalla tabella di verità risulta sempre possibile giungere alle cosiddette forme canoniche. Le forme canoniche ci consentono di scrivere la forma algebrica a partire dalla tabella di verità. Esse hanno valore concettuale ma, come vedremo, minimo valore pratico.

A.2.1 Prima forma canonica

Data la tabella di verità di una funzione f di n variabili $(x_1, x_2, \ldots, x_n)$, la prima forma canonica si ottiene come somma di un numero di termini pari al numero di righe in cui la funzione vale 1, ciascuno dei quali è costituito dal prodotto di tutte le variabili $(x_1, x_2, \ldots, x_n)$; nel prodotto, ciascuna variabile appare in forma diretta o complementata a seconda del fatto che, sulla corrispondente riga su cui la funzione vale 1, la variabile valga 1 o 0. Un tale termine viene denominato *prodotto fondamentale*[4].

La funzione di Figura A.5 che qui riportiamo:

$$y = f(x_1, x_2, x_3) = \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 \quad \text{(A.1)}$$

è la prima forma canonica, corrispondente alla tabella ivi riportata. Per il modo in cui è costruita, ogni termine della somma contribuisce a rendere 1 la funzione soltanto per la configurazione delle variabili ad esso corrispondente. Ad esempio, il termine $\overline{x}_1 \cdot x_2 \cdot \overline{x_3}$ rende vera la funzione solo per la configurazione 010. La funzione in esame vale 1 sulla terza, quinta, sesta, settima e ottava riga, corrispondenti alla numerazione 2, 4, 5, 6 e 7. Pertanto, un modo sintetico per rapresentare la precedente espressione di y è il seguente:

$$y = \sum_{3}(2, 4, 5, 6, 7)$$

Il significato è intuitivo: y è una funzione di 3 variabili, data dalla somma dei mintermini di peso 2, 4, 5, 6 e 7.

A.2.2 Seconda forma canonica

Data la tabella di verità di una funzione f di n variabili $(x_1, x_2, \ldots, x_n)$, la seconda forma canonica si ottiene come prodotto di un numero di termini pari al numero di righe in cui la funzione vale 0, ciascuno dei quali è costituito dalla somma di tutte le

[4]La prima forma canonica viene anche detta *somma di prodotti fondamentali*, ovvero *somma di mintermini.*

variabili $(x_1, x_2, \ldots, x_n)$, in forma diretta o complementata a seconda del fatto che nella corrispondente riga su cui la funzione vale 0 la variabile valga 0 o 1. I termini della somma vengono detti *somme fondamentali*[5].

Alla forma tabellare di Figura A.5 corrisponde la seguente (seconda) forma canonica:

$$y = (x_1 + x_2 + x_3) \cdot (x_1 + x_2 + \overline{x_3}) \cdot (x_1 + \overline{x_2} + \overline{x_3}) = \prod_3 (0, 1, 3)$$

A.2.3 Proprietà delle forme canoniche

Le forme canoniche danno luogo a reti a due livelli. Per la Somma di Prodotti (SP) il livello di uscita è una porta OR, il livello di ingresso è costituita di sole porte AND i cui ingressi possono anche essere in forma complementata. Per il Prodotto di Somme (PS) il livello di uscita è una porta AND, il livello di ingresso è di sole porte OR[6].

La seconda forma canonica può essere ottenuta dalla prima e viceversa ricorrendo al teorema di De Morgan (si veda l'Esercizio A.3). Il passaggio è immediato ricorrendo alla notazione concisa: O

$$y = \textstyle\sum_3 (2, 4, 5, 6, 7) = \prod_3 (0, 1, 3)$$

Ovviamente, data una forma canonica in modo algebrico, si può passare alla duale

Qualsiasi espressione può sempre essere ricondotta in forma canonica. Si fornisce un esempio esplicativo partendo dalla seguente funzione in forma SP[7]:

$$f(x_1, x_2, x_3) = \overline{x}_1 \cdot x_2 + x_1 \cdot x_2$$

questa può essere riportata alla prima forma canonica, moltiplicando ciascun prodotto per $(a + \overline{a})$, essendo a la variabile che non compare nel prodotto:

$$\begin{aligned} f(x_1, x_2, x_3) &= (\overline{x}_1 \cdot x_2) \cdot (\overline{x}_3 + x_3) + (x_1 \cdot x_2) \cdot (\overline{x}_3 + x_3) \\ &= \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 + \overline{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x}_3 + x_1 \cdot x_2 \cdot x_3 \end{aligned}$$

Avvertenza

D'ora in avanti faremo quasi esclusivamente riferimento alle sole espressioni in forma SP, in quanto esse sono più agevoli da manipolare. Il principio di dualità ci garantisce che una proprietà valida per le somme di prodotti è valida in modo duale anche per i prodotti di somme. Inoltre, d'ora in avanti, il segno di prodotto logico $(\cdot)$ verrà omesso, a meno di non generare ambiguità.

---

[5] La seconda forma canonica viene anche detta *prodotto di somme fondamentali*, ovvero *somma di maxtermini*.

[6] Si noti che, nel considerare le forme SP e PS come reti a due livelli, non si tiene conto degli eventuali negatori in ingresso. Ciò è del tutto ragionevole, in quanto, nella pratica dei sistemi digitali, i segnali sono normalmente disponibili in forma sia diretta sia complementata.

[7] Ovviamente, se la funzione è in forma PS si procede in modo duale.

A.2.4 Minimizzazione

Riprendiamo la funzione in forma canonica (A.1)

$$y = \overline{x}_1 x_2 \overline{x}_3 + x_1 \overline{x}_2\, \overline{x}_3 + x_1 \overline{x}_2 x_3 + x_1 x_2 \overline{x}_3 + x_1 x_2 x_3$$

e domandiamoci se è possibile semplificarla. Per motivi che saranno subito chiari, conviene aggiungere, duplicandolo, l'ultimo termine della somma. Si ottiene

$$y = \overline{x}_1 x_2 \overline{x}_3 + x_1 \overline{x}_2 \overline{x}_3 + x_1 \overline{x}_2 x_3 + x_1 x_2 \overline{x}_3 + x_1 x_2 x_3 + x_1 x_2 x_3$$

Ora, in base alla proprietà distributiva, il primo e l'ultimo termine possono essere così semplificati: $\overline{x}_1 x_2 \overline{x}_3 + x_1 x_2 x_3 = (\overline{x}_1 + x_1) x_2 \overline{x}_3 = x_2 x_3$. Analogamente, il secondo e terzo termine si semplificano in $x_1 \overline{x}_2$, mentre il quarto e quinto termine si semplificano in $x_1 x_2$. Ovvero si ha

$$y = x_2 \overline{x}_3 + x_1 \overline{x}_2 + x_1 x_2$$

che si semplifica ulteriormente in

$$y = x_1 + x_2 \overline{x}_3 \tag{A.2}$$

È evidente che, da un punto di vista pratico, la rete corrispondente alla forma canonica da cui siamo partiti è meno conveniente di quella corrispondente a (A.2). Infatti, la prima si realizza con una porta OR a cinque ingressi e cinque porte AND a tre ingressi, mentre la seconda richiede una sola porta OR e una sola porta AND ambedue a due ingressi.

In generale si pone il problema della minimizzazione. Ovvero di trovare l'espressione minima per una data funzione logica. Nell'esempio appena fatto la semplificazione è stata ottenuta applicando le regole dell'algebra. Ciò ha richiesto, all'inizio, l'aggiunta di un termine ai fini della successiva semplificazione. L'applicazione delle regole dell'algebra in certe situazioni può rivelarsi piuttosto ostica e non è detto che si pervenga sempre alla forma ottima, quella che prevede il minor numero di porte e, a parità di numero di porte, quella con porte con minori ingressi. Conviene cercare un metodo che dia la garanzia del raggiungimento della espressione minima. Qui si seguito ci applicheremo al solo caso di reti SP, lasciano al lettore volenteroso il caso delle reti PS.
Il metodo si basa sul fatto che, data l'espressione $A\overline{a} + Aa$, dove A è un qualsiasi prodotto di termini e a è una variabile booleana, essa si può semplificare raccogliendo A a fattor comune, ottenendo:

$$A\overline{a} + Aa = A(\overline{a} + a) = A \cdot 1 = A \tag{A.3}$$

Si dice che le espressioni $A\overline{a}$ e Aa sono adiacenti, perché le configurazioni delle variabili che le individuano hanno distanza unitaria, ovvero differiscono solo per la variabile che compare in forma diretta e in forma complementata. La minimizzazione delle funzioni SP consiste nell'individuazione delle configurazioni adiacenti e nell'applicazione sistematica della precedente proprietà. Le mappe di Karnaugh [KJ10], [MK08]sono lo strumento normalmente usato quando il numero delle variabili non è superiore a 5.

A.2.5 Mappe di Karnaugh

Le tabelle di verità non permettono di individuare i termini adiacenti. Per questo si ricorre alle mappe di Karnaugh. Queste non sono altro che una rappresentazione conveniente delle tabelle di verità: ogni casella sulla mappa corrisponde ad una riga della

tabella e in essa si scrive 1 o 0, a seconda del valore che prende la funzione per la configurazione delle variabili che danno le coordinate della casella nella mappa. In Figura A.7 sono riportate mappe di 1, 2 e 3 variabili.

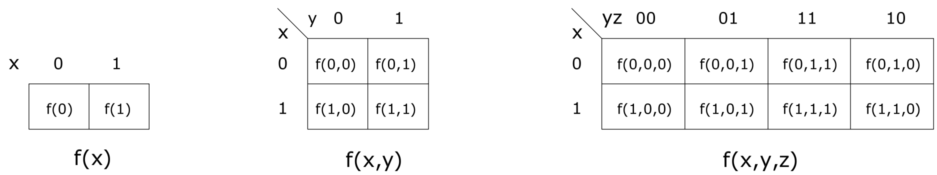


Figura A.7 Mappe di Karnaugh di ordine 1, 2 e 3. La figura indica chiaramente che ogni casella riporta il valore di f per la configurazione delle variabili che ne dà le coordinate.

Per capire il sistema di coordinate delle mappe, conviene fare il seguente ragionamento geometrico. Data una funzione di n variabili, le 2^n configurazioni delle variabili possono essere viste come le coordinate dei 2^n vertici di un (iper)cubo di ordine n (n-cubo), con spigoli di lunghezza unitaria, sistemato con un vertice nell'origine degli assi cartesiani di uno spazio a n-dimensioni e con gli spigoli che escono da tale vertice a giacere sugli assi stessi. Due vertici adiacenti differiscono per una sola coordinata, dunque il vertice posto nell'origine (0,0,...,0) ha come adiacenti tutti i vertici sistemati sugli assi cartesiani e cioè i vertici di coordinate (1,0,...,0), (0,1,...,0),..., (0,0,...,1). Il vertice (1,1,...,1) risulta opposto all'origine.

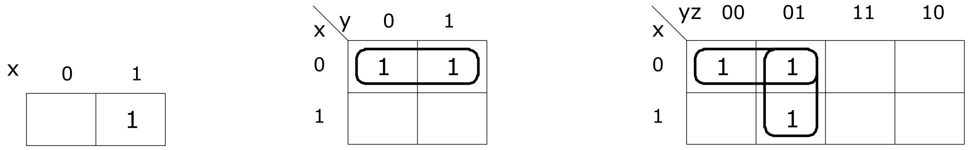
Per semplicità si faccia riferimento ad una funzione di tre variabili e si immagini di aprire il cubo e stendere sul piano le sue facce, mantenendo per ciascun vertice l'indicazione delle sue coordinate. Se ora si immagina di dilatare i punti che corrispondono ai vertici e trasformarli in caselle, si ottiene una mappa di Karnaugh di ordine 3, per la quale, se la disposizione è in orizzontale, ne risultano le coordinate come nella mappa a destra di Figura A.7.

Caselle adiacenti differiscono per una sola coordinata. In Figura A.7 la casella (0,0,0) è adiacente alle 3 caselle (0,0,1), (0,1,0) e (1,0,0). Se su due caselle adiacenti la funzione vale 1, allora si ha una una situazione come quella dell'espressione (A.3) e quindi il contributo delle due caselle equivale al prodotto delle variabili che non cambiano nel passare da una casella all'altra.

Trattando della forma SP è conviene riportare solamente gli 1, come in Figura A.8, nella quale si hanno le mappe di tre funzioni; considerando la mappa di ordine due di Figura A.8, la forma canonica è: $f(x,y) = \overline{x}\,\overline{y} + \overline{x}y$, che si semplifica in $f(x,y) = \overline{x}$. Per la mappa di destra la forma canonica è: $f(x,y,z) = \overline{x}\,\overline{y}\,\overline{z} + \overline{x}\,\overline{y}z + x\overline{y}z$, che si semplifica in $f(x,y,z) = \overline{x}\,\overline{y} + \overline{y}z$. Sulle tabelle di Figura A.8 sono stati individuati i raggruppamenti di 1 corrispondenti ai prodotti di variabili che compaiono nella forma semplificata delle funzioni. Come detto qui di seguito tali raggruppamenti vengono detti *sottocubi*.

I concetti precedentemente esposti in riferimento a casi particolari vengono così generalizzati:

- una funzione booleana di n variabili $f(x_1, x_2, ..., x_n)$ viene rappresentata su una mappa di Karnaugh di ordine n;
- una mappa di ordine n contiene 2^n celle;



$f(x) = x$ $\qquad f(x,y) = \overline{x}$ $\qquad f(x,y,z) = \overline{x}\,\overline{y} + \overline{y}\,z$

Figura A.8 Esempi di funzioni sulle mappe di Karnaugh e loro minimizzazione.

- le coordinate delle 2^n celle corrispondono alle 2^n possibili configurazioni delle n variabili;
- sulla mappa (di ordine n) le celle sono disposte in modo tale per cui ogni cella è adiacente a n celle;
- su una mappa di ordine n si definisce *sottocubo* (o sottomappa) di ordine m, con $m \leq n$, un insieme di 2^m celle tale per cui ciascuna casella del sottocubo è adiacente a m caselle del sottocubo stesso;
- se su tutte le celle di un sottocubo di ordine m la funzione vale 1, il contributo complessivo delle 2^m celle è dato dal prodotto delle $(n - m)$ variabili che non variano nel sottocubo. Nel prodotto la variabile compare in forma diretta se nel sottocubo vale 1, in forma complementata se nel sottocubo vale 0.

Si definisce *copertura* della funzione un insieme di sottocubi tale da *coprire* tutti gli 1 della funzione stessa. La minimizzazione consiste nel trovare una copertura formata da un insieme di sottocubi, ciascuno dei quali sia il più ampio possibile e non sia contenuto in altri sottocubi.

Come esempio si consideri la funzione di tre variabili riportate nelle mappe di Figura A.9. Evidentemente sia l'espressione a) che l'espressione b) rappresentano la funzione data. La seconda, costruita seguendo il criterio di espandere per quanto possibile i sottocubi di copertura degli 1, costituisce l'espressione minima.

Per completezza si deve ricordare che si definisce *implicante* il prodotto di variabili corrispondente ad un sottocubo in cui la funzione vale 1. Un implicante si dice *primo* se corrisponde ad un sottocubo non completamente coperto da un altro sottocubo su cui la funzione è 1. Il problema della minimizzazione di un funzione booleana in forma SP può essere dunque riformulato come quello della ricerca di un insieme di implicanti primi che coprono la funzione. In Figura A.9 gli implicanti $\overline{x}$ e $\overline{y}$ sono primi, mentre non lo sono $\overline{x}\,\overline{z}$, $\overline{y}z$ e $\overline{x}y$.

Sulla mappa di destra di Figura A.10 si notino quattro sottocubi di ordine 1, ciascuno dei quali copre in modo esclusivo un 1 della funzione. I corrispondenti implicanti primi si dicono *essenziali.* Gli implicanti essenziali fanno necessariamente parte della copertura minima nella funzione. L'implicante primo xz di figura, sebbene corrisponda ad un sottocubo di ordine 2, non entra a far parte delle copertura ottima, in quanto non essenziale.

Per dualità, le mappe di Karnaugh si possono impiegare anche per minimizzare le funzioni in forma di Prodotti di Somme (PS). A tal fine, si raggruppano le caselle contenenti il valore 0 e si associa ad ogni raggruppamento la somma delle variabili che non variano sul raggruppamento, prendendole in forma complementata se appaiono come 1 nel raggruppamento. Per esempio, se si prende la funzione riportata nella mappa ordine



a) $f(x,y,z) = \overline{y}\,\overline{z} + \overline{y}\,z + \overline{x}\,y$ *b*) $f(x,y,z) = \overline{x} + \overline{y}$

Figura A.9 Esempio di due diverse coperture di un stessa funzione. La copertura di destra, essendo formata da sottocubi più ampi, fornisce la minima espressione SP.

tre a destra in Figura A.8 si ha un sottocubo di ordine 2 e uno di ordine 1, i quali danno luogo alla minima espressione PS: $f(x,y,z) = \overline{y}(\overline{x} + z)$. In questo caso, effettuando il prodotto si ottiene la forma minima SP: $f(x,y,z) = \overline{x}\ \overline{y}\ +\ \overline{y}z$.

A.2.6 Metodi algoritmici

La tecnica delle mappe è praticamente inapplicabile quando il numero delle variabili supera 5 a causa della difficoltà di rappresentazione. La ricerca dell'espressione minima può essere automatizzata con il metodo di Quine-McCluskey [McC56]. Tuttavia la complessità cresce in modo esponenziale con il numero degli ingressi. Se il numero di ingressi supera la ventina la minimizzazione tende a diventare un problema intrattabile. I sistemi per la progettazione automatica accoppiano il metodo di Quine-McCluskey con tecniche euristiche, in modo da pervenire a soluzioni accettabili anche se non necessariamente ottime.



a) $f(w,x,y,z) = \overline{w}\,\overline{z} + wy\overline{z} + \overline{x}\,\overline{z}$ *b*) $f(w,x,y,z) = \overline{w}x\overline{y} + \overline{w}yz + wxy + w\overline{y}z$

Figura A.10 Esempio di mappe di ordine 4. Sulla mappa di sinistra si noti l'implicante corrispondente alle 4 caselle poste agli angoli della mappa. Esso porta un contributo pari a $\overline{xy}$. L'espressione minima SP relativa alla mappa di destra non comprende l'implicante primo xz in quanto non essenziale.

A.2.7 Funzioni non completamente specificate – Condizioni di indifferenza

Molto frequentemente, le specifiche di una rete logica definiscono il comportamento solo per un sottoinsieme delle possibili configurazioni di ingresso.

Prendiamo per esempio la realizzazione di una rete per la decodifica da codice BCD a codice Eccesso 3. I codici BCD ed Eccesso 3 sono definiti su 4 bit. Il codice BCD corrisponde alla codifica binaria delle dieci cifre decimali {0,1,...,9} (Appendice B), mentre i codici Eccesso 3 corrispondono ordinatamente ai codici BCD incrementati di 3. Detti A, B, C, D gli ingressi BCD e w, x, y, z le uscite Eccesso 3, la tabella di verità è mostrata in Figura A.11. Si noti che sulle righe non corrispondenti a codici BCD, è stato riportato il segno "–". Tale segno indica una *condizione di indifferenza*, ovvero una condizione per la quale la funzione di uscita non è definita ed è perciò irrilevante il fatto che la funzione assuma valore 0 o 1.

Se il valore della funzione in corrispondenza alle condizioni di indifferenza è irrilevante, allora queste possono essere utilizzate nel modo più conveniente per ottenere la copertura della funzione stessa. In pratica si tratta di utilizzare le condizioni di indifferenza per allargare al massimo i sottocubi di copertura della funzione. Seguendo questo criterio si ottengono le coperture riportate in Figura A.11, cui corrispondono le seguenti funzioni per w, x, y e z:

$$w = A + B\ D + B\ C \qquad x = B\ \overline{C}\ \overline{D} + \overline{B}\ D + B\ C$$
$$y = \overline{C}\ \overline{D} + C\ D \qquad z = \overline{D}$$

È ovvio che, scelta una copertura, questa determina in modo univoco i valori delle uscite, anche per le configurazioni di ingresso corrispondenti a condizioni di indifferenza. Se richiesto, spetta al progettista prendere i provvedimenti del caso quando si presentino tali configurazioni.

A.3 Altri operatori e altri tipi di porta

Mediante il teorema di De Morgan è possibile dimostrare che qualsiasi espressione logica può essere espressa tramite due soli tipi di operatori; infatti, somma e prodotto logico possono essere decomposti nel seguente modo:

$$x + y = \overline{\overline{x + y}} = \overline{\overline{x} \cdot \overline{y}} \qquad\qquad x \cdot y = \overline{\overline{x \cdot y}} = \overline{\overline{x} + \overline{y}}$$

In conclusione, mediante le sole due operazioni di prodotto e complementazione, oppure tramite le sole due operazioni di somma e complementazione, si è in grado di realizzare qualsiasi espressione. Formalmente, i precedenti concetti si esprimono nel modo seguente:

- l'insieme degli operatori algebrici $\{+, \cdot, ^{-}\}$ è funzionalmente ridondante;
- l'insieme $\{+, ^{-}\}$ e l'insieme $\{\cdot\ , ^{-}\}$ sono funzionalmente completi.

A.3.1 NAND e NOR

Introduciamo ora due ulteriori operatori funzionalmente completi: l'operatore NAND e l'operatore NOR.

A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	-	-	-	-
1	0	1	1	-	-	-	-
1	1	0	0	-	-	-	-
1	1	0	1	-	-	-	-
1	1	1	0	-	-	-	-
1	1	1	1	-	-	-	-

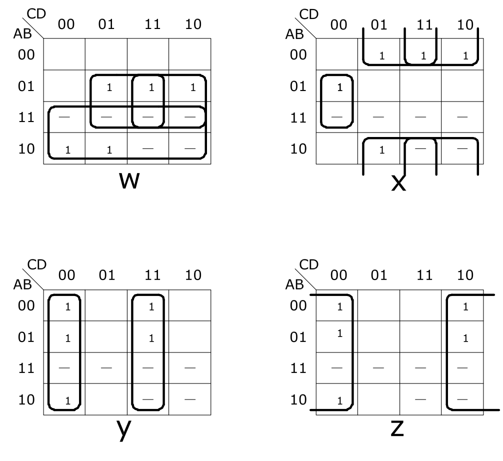


Figura A.11 Tabella di decodifica da codice BCD a Eccesso 3 e mappe e coperture delle funzioni di uscita. I trattini indicano condizioni di indifferenza.

L'operatore NAND consiste nella complementazione del prodotto e si indica con una barra verticale tra gli operandi ($x_1 \,|\, x_2$), mentre la porta NAND si disegna giustapponendo il cerchietto di negazione sull'uscita della porta AND.

L'operatore NOR consiste nella complementazione della somma e si indica con una freccia verticale rivolta verso il basso tra gli operandi ($x_1 \downarrow x_2$), mentre la porta NOR si disegna giustapponendo il cerchietto di negazione sull'uscita della porta OR.

In Figura A.12 vengono riportati i simboli corrispondenti ai due operatori e le relative tabelle di verità.



x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

x	y	x
0	0	1
0	1	0
1	0	0
1	1	0

Figura A.12 Simboli delle porte NAND e NOR e relative tabelle di verità.

Le porte NAND e NOR permettono di realizzare qualunque funzione logica utilizzando un solo tipo di porta, in quanto sia NAND sia NOR sono *funzionalmente completi*, come si mostra facilmente con il teorema di De Morgan:

Complementazione

$$\overline{x} = \overline{x \cdot x} = x \,|\, x \qquad\qquad \overline{x} = \overline{x + x} = x \downarrow x$$

Somma

$$x_1 + x_2 = \overline{\overline{x_1 + x_2}} = \overline{\overline{x}_1 \cdot \overline{x}_2} = (\overline{x}_1 \,|\, \overline{x}_2)$$

$$x_1 + x_2 = \overline{\overline{x_1 + x_2}} = \overline{(x_1 \downarrow x_2)} = \overline{(x_1 \downarrow x_2) + (x_1 \downarrow x_2)} = (x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2)$$

Prodotto

$$x_1 \cdot x_2 = \overline{\overline{x_1 \cdot x_2}} = \overline{(x_1 \,|\, x_2)} = \overline{(x_1 \,|\, x_2)(x_1 \,|\, x_2)} = (x_1 \,|\, x_2) \,|\, (x_1 \,|\, x_2)$$

$$x_1 \cdot x_2 = \overline{\overline{x_1 \cdot x_2}} = \overline{\overline{x}_1 + \overline{x}_2} = (\overline{x}_1 \downarrow \overline{x}_2)$$

Le precedenti espressioni vengono riportate in forma schematica in Figura A.13.

Si osservi che per gli operatori NAND e NOR non vale la proprietà associativa. Infatti, preso per esempio per il NAND:

$$a \,|\, b \,|\, c = \overline{abc} = \overline{a} + \overline{b} + \overline{c} \neq (a \,|\, b) \,|\, c = \overline{\overline{(ab)}c} = ab + \overline{c}$$

L'uso dei due operatori risulta, dunque, piuttosto ostico. Per questo motivo, anche quando si vuole arrivare a reti con un solo operatore, si preferisce ragionare in termini di AND, OR e NOT ed effettuare la trasformazione in rete di soli NAND (NOR) come ultimo passo.



Figura A.13 Costruzione delle operazioni di NOT, OR e AND dalle porte NAND e NOR.

A.3.2 Reti con sole porte NAND o sole porte NOR

Il passaggio da espressioni in forma SP (PS) a reti NAND (NOR) risulta immediato tramite il teorema di De Morgan. Ad esempio:

$$z = a + bc + de = \overline{\overline{a + bc + de}} = \overline{\overline{a}\,\overline{(bc)}\,\overline{(de)}} = \overline{a}|(b|c)|(d|e)$$

$$z = (a+b)(c+d)e = \overline{\overline{(a+b)(c+d)e}} = \overline{\overline{(a+b)} + \overline{(c+d)} + \overline{e}} = (a \downarrow b) \downarrow (c \downarrow d) \downarrow \overline{e}$$

Per la forma SP tutto si riduce a sostituire gli operatori AND e OR con NAND. Per la forma PS i medesimi operatori vengono sostituiti con NOR. In Figura A.14 si mosta il passaggio per il caso SP. Notare che il termine che entra direttamente nella porta di uscita compare complementato nelle precedenti espressioni, ovvero passato attraverso una porta NAND o NOR rispettivamente.

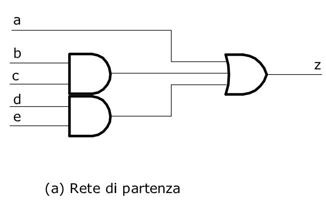


Figura A.14 Passaggio da rete SP ($z = a + bc + de$) a rete NAND ($z = \overline{a}|(b|c)|(d|e)$). Il passaggio richiede solo la sostituzione di porte AND e OR con porte NAND.

Rappresentazione alternativa

In base al teorema di De Morgan, le porte NAND e NOR possono essere anche rappresentate con i simboli alternativi di Figura A.15. Questi simboli alternativi risultano utili nel passaggio da reti AND/OR a reti NAND o NOR, aggiustando i *fig:reti:pallini* di complementazione sulle reti AND/OR, in modo da far comparire le porte NAND o NOR. Il procedimento è in due passi:

1) si inseriscono, sui rami che collegano le uscite delle porte di primo livello AND (OR) con gli ingressi della porta di secondo livello OR (AND); coppie di negatori.
2) si sostituisce il simbolo della porta di secondo livello con quello usuale del NAND (NOR).



Figura A.15 Simboli equivalenti per le porte NAND e NOR

In Figura A.16 la tecnica viene mostrata in riferimento alla rete in forma PS corrispondente a $z = (a + b)(c + d)e$.

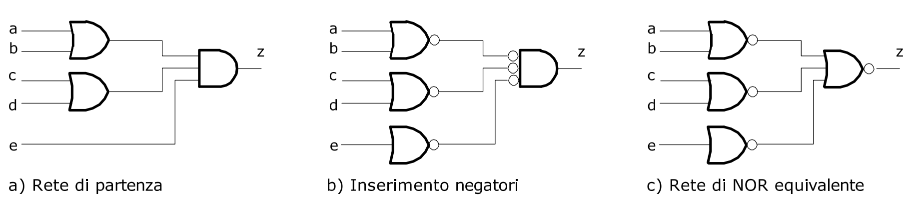


Figura A.16 Passaggio da rete SP ($z = (a + b)(c + d)e$) a rete di NOR ($z = (a \downarrow b) \downarrow (c \downarrow d) \downarrow \overline{e}$).

A.3.3 XOR e NXOR

L'operatore XOR (ovvero OR esclusivo, rappresentato come $\oplus$) e il suo negato NXOR sono molto utilizzati nella pratica per snellire la rappresentazione algebrica e grafica di funzioni logiche. I simboli usati per queste porte e le relative tabelle di verità sono riportati in Figura A.17.



x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

x	y	z
0	0	1
0	1	0
1	0	0
1	1	1

Figura A.17 Simboli delle porte XOR e NXOR e relative tabelle di verità.

Lo XOR dà risultato vero se e solo se $x_1 \neq x_2$. In termini algebrici esso è così definito:

$$y = x_1\overline{x_2} + \overline{x_1}x_2 = x_1 \oplus x_2$$

Per l'operatore XOR è valida la proprietà associativa. La Figura A.18, a sinistra, riporta la costruzione dell'operatore XOR con sole porte NAND; la rete è stata ottenuta applicando il metodo descritto al Paragrafo A.3.2. A destra viene riportata la realizzazione ottima in termini di soli NAND;

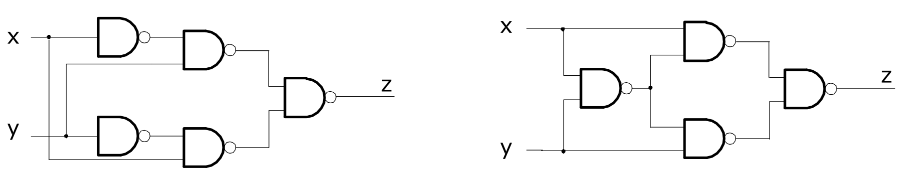


Figura A.18 Realizzazione di XOR come rete di soli NAND. A destra la realizzazione ottima.

L'operatore NXOR ($\equiv$) dà risultato vero se e solo se gli ingressi sono uguali. Per questo viene anche detto "identità" o "equivalenza". In termini algebrici è così definito:

$$y = \overline{x}_1\,\overline{x}_2 + x_1x_2 = x_1 \equiv x_2$$

A.4 Qualche osservazione sulle porte logiche

Non è scopo di questo libro trattare questioni tecnologiche, tuttavia è bene illustrare alcuni aspetti fondamentali delle tecnologie dei circuiti integrati.

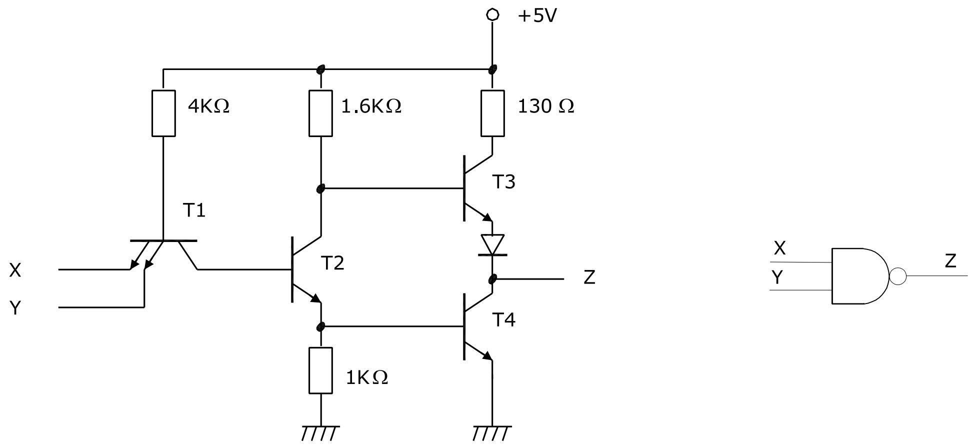


Figura A.19 Tipica porta TTL con uscita *totem pole* e corrispondente porta logica. Lo schema corrisponde ad una delle quattro porte NAND contenute nell'integrato SN7400.

In Figura A.19 viene mostrato lo schema classico della porta NAND a due ingressi, in logica TTL. La logica TTL (*Transistor-Transistor Logic*), nota anche come Serie SN74, è stata per lungo tempo dominante nel mondo dell'elettronica digitale. La serie comprende svariate centinaia di dispositivi, identificati attraverso un numero. Il primo componente della serie è l'integrato 7400, un dispositivo SSI (*Small Scale Integration*) contenente quattro porte NAND.

Sebbene ormai in disuso, la Serie SN74 continua a rappresentare il termine di paragone e di riferimento, quanto a funzionalità dei componenti, per le famiglie logiche successivamente prodotte, anche in altre tecnologie, compresa la tecnologia CMOS. Per tale motivo, anche in questo libro, si fa normalmente riferimento a questa serie.

Gli aspetti caratteristici del circuito di Figura A.19 sono il transistore multiemettitore in ingresso e lo stadio di uscita che, per la sua conformazione, viene indicato figuratamente come *totem pole*. Quando ambedue gli ingressi sono a livello alto il transistore T4 si trova in saturazione (uscita y bassa), mentre se almeno uno dei due ingressi è a livello basso il transistore T4 risulta interdetto (uscita y alta). In logica positiva questo comportamento è quello di una porta NAND.

Lo stadio di uscita a totem pole è dotato di elevata velocità di commutazione e buona immunità al rumore, in quanto è in grado di far passare corrente in ambedue le direzioni attraverso percorsi a bassa impedenza.

In Figura A.1 è stato indicato che i livelli di tensione di una rete binaria devono mantenersi al di sopra di V_{Amin} in stato di alto e al di sotto di V_{Bmax} in stato di basso. In realtà per la logica TTL (e per tutte le altre) le soglie sono differenti a seconda che si tratti di terminali di ingresso o di uscita delle porte. Ciò garantisce che, in presenza di un accettabile disturbo, la tensione di uscita di una porta non entra nella fascia proibita per l'ingresso di una porta ad essa collegata. La differenza rappresenta il cosiddetto *margine di rumore*. Per la TTL standard le soglie sono quelle di Figura A.20.

Come si vede, il margine di rumore è circa 0,4V, sia in stato alto sia in stato basso. Quella di Figura A.20 rappresenta la situazione di caso peggiore, perché i valori tipici di tensione assunti dalla logica TTL sono 3,4V in stato alto e 0,3V in stato basso. Nella

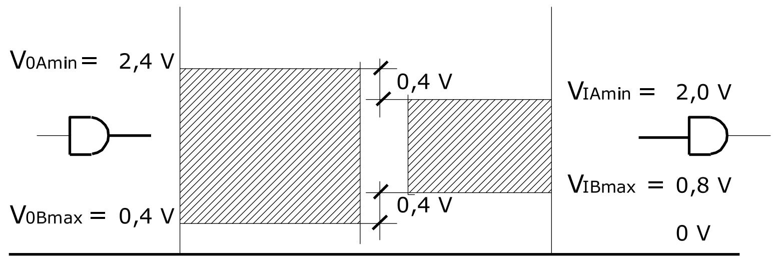


Figura A.20 Soglie e margine di rumore per la logica TTL standard. V_{OAmin} e V_{IAmin} sono le soglie in stato alto, V_{OBmax} e V_{IBmax} sono le soglie in stato basso, per uscite e ingressi rispettivamente.

TTL standard il tempo di commutazione da alto a basso è di circa 7 ns e di 10 ns da basso a alto.

In Figura A.21 viene invece schematizzato il comportamento di una porta TTL per quanto riguarda la capacità di pilotare altre porte. I_0 indica la corrente *massima garantita* al morsetto di uscita nei due stati. Come si vede, in stato logico 1 (Alto) la corrente è uscente e vale 400 μA; in stato logico 0 (Basso) la corrente è entrante e vale 16 mA. Nello stato 1 un ingresso TTL assorbe al massimo 40 μA, mentre in stato 0 un ingresso TTL eroga al massimo 1,6 mA. Da ciò consegue che sia in stato alto sia in stato basso un'uscita TTL può pilotare fino a 10 ingressi TTL. Ovvero che è pari a 10 il *fan-out*. Naturalmente se a valle di una porta TTL sono collegati ingressi di un'altra famiglia logica, il numero massimo di ingressi pilotabili dipende dalle caratteristiche di assorbimento/erogazione degli ingressi collegati.

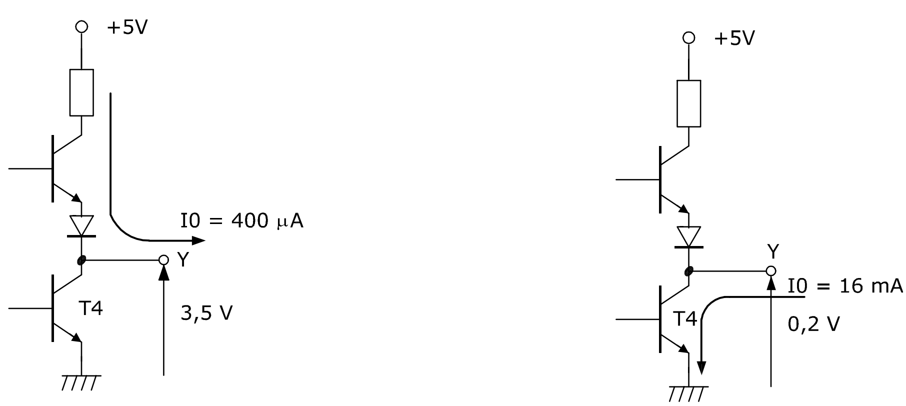


Figura A.21 Correnti massime (garantite) in uscita nello stato logico 1 e nello stato logico 0 di una porta TTL. In stato alto il transistore T4 è interdetto e la corrente fluisce da Vcc (+5 V) verso gli ingressi collegati a valle (è garantito che la porta eroghi fino a 400 μA); in stato basso la corrente fluisce dagli ingressi collegati a valle verso massa attraverso il transistore T4 in saturazione (è garantito un assorbimento fino a 16 mA).

Uscita a collettore aperto.

La logica con uscita totem pole non risulta adatta al collegamento di due o più uscite

su di una stessa linea comune. Essa è fatta per collegare le uscite agli ingressi di porte a valle. Il collegamento di due uscite può determinare una situazione di cortocircuito con conseguente danneggiamento dei dispositivi.

In Figura A.22 viene dato lo schema di una porta TTL con uscita a *collettore aperto* (nel caso di logica MOS si parla di *Open-Drain*). Il nome deriva dal fatto che la resistenza di *pull-up* è esterna al dispositivo. Un dispositivo a collettore aperto ha l'uscita in stato basso quando il transistore dello stadio di uscita è in saturazione, ha l'uscita in stato alto quando il transistore è in interdizione. Più uscite a collettore aperto possono condividere la stessa linea di collegamento ed è consentito che una o più uscite siano attive (stato basso) e le altre in stato alto.

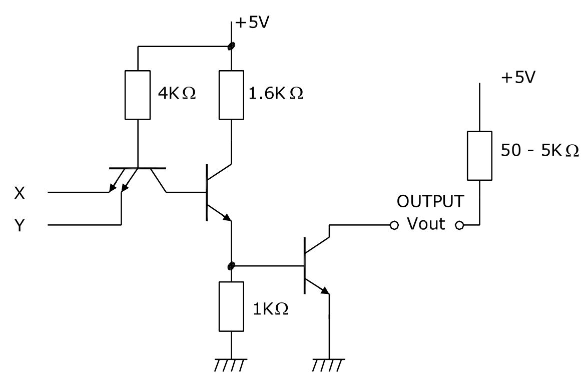


Figura A.22 Porta TTL con uscita a collettore aperto. La resistenza di *pull-up* è esterna al dispositivo stesso.

L'uscita a collettore aperto realizza il cosiddetto AND cablato (*Wired-AND*) in logica positiva. Infatti, se più uscite a collettore aperto sono collegate sulla stessa linea, è sufficiente che una sola sia a livello basso affinché la linea sia a livello basso[8]. L'impiego di porte con uscita a collettore aperto è illustrato in Figura A.23.

Il vantaggio della logica a collettore aperto non sta solo nel risparmio della porta AND (si veda Figura A.23), quanto nella maggior flessibilità costruttiva che deriva dal suo impiego nella realizzazione di strutture a bus (Paragrafo A.11.1). Facendo riferimento alla Figura A.23, si supponga che per qualche motivo la rete debba prevedere una ulteriore porta NAND oltre alle due esistenti e che pure la sua uscita debba essere legata in AND con le altre. Con la logica a collettore aperto questo AND si ottiene collegando direttamente l'uscita della porta aggiunta al punto a comune delle due uscite esistenti. Da un punto vista costruttivo, si evita di intervenire sul circuito provvedendo un contatto su cui attestare tutte le eventuali uscite da mettere in AND. Ciò conduce in modo naturale alle strutture a *bus* descritte al Paragrafo A.11.1. Al contrario, con uscite in totem pole, per ottenere l'AND non resta che provvedere una porta a tre ingressi in luogo dell'esistente, con la necessità di intervenire fisicamente sul circuito[9]

[8]In considerazione del fatto che su una linea a collettore aperto lo stato di *non attivo* è quello alto, può essere conveniente ragionare in termini di logica negativa e parlare di OR cablato (*Wired-OR*).

[9]Le modalità possono essere diverse, ma richiedono comunque un qualche intervento sul circuito.

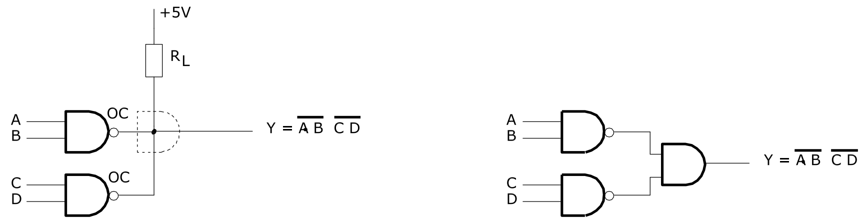


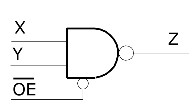
Figura A.23 La parte sinistra mostra l'impiego di porte con uscita a collettore aperto per la realizzazione dell'AND cablato. A destra viene mostrata la rete equivalente realizzata con porte con uscita standard (totem pole).

Uscita in terzo stato

A metà degli anni '70 viene introdotta la logica con uscita a tre stati (*tristate*). Essa ha il seguente funzionamento: tramite un ingresso di controllo separato, vengono portati in interdizione entrambi i transistori dell'uscita totem pole; l'uscita viene così a trovarsi in un terzo stato ad alta impedenza e tutto avviene come se essa non fosse collegata. La tabella di verità della porta è riportata in Figura A.24. Usando questo tipo di logica, più dispositivi possono condividere con la loro uscita una medesima linea, con il vincolo di avere al più un dispositivo con l'uscita in conduzione e tutti i rimanenti con l'uscita in terzo stato (Figura A.25). A differenza del caso di logica a collettore aperto, non si rende necessaria alcuna resistenza di pull-up sulla linea a comune.

L'impiego di questa tecnologia è diventato diffusissimo e rappresenta lo standard nel mondo dei microprocessori, memorie ecc., dove normalmente più componenti presentano le loro uscite sul medesimo bus in tempi diversi. Il motivo del successo della logica con uscita a tre stati è da ricercarsi nel fatto che essa fornisce il meglio dei due mondi e cioè:

a) l'alta velocità di commutazione dell'uscita totem pole;
b) la capacità di collegamenti multipli dell'uscita a collettore aperto.



x	y	OE	z
0	0	1	1
0	1	1	0
1	0	1	0
1	1	1	0
–	–	0	ϕ

Figura A.24 Simbolo logico e tabella di verità di una porta NAND con uscita in terzo stato. Il simbolo ϕ corrisponde al terzo stato di alta impedenza.

A.5 Notazione per i segnali

L'impiego dei simboli 0 e 1 è molto conveniente quando si devono manipolare espressioni logiche, ma a volte può oscurare la comprensione del fenomeno fisico. Si consideri, ad esempio, il segnale di controllo che abilita l'uscita di una porta in logica tre stati. È possibile che il segnale svolga la sua funzione nello stato H (Alto), come pure nello stato

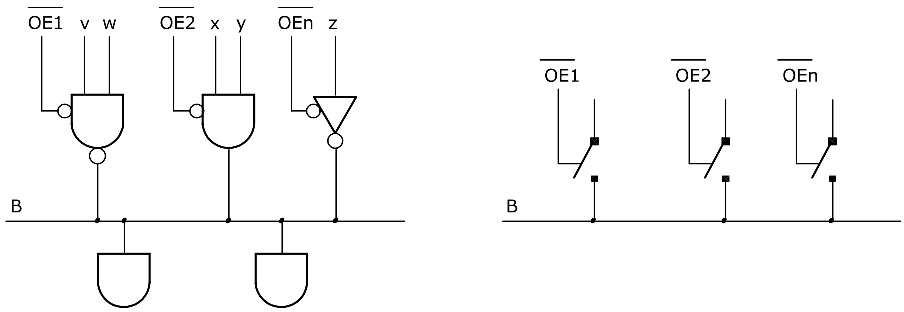


Figura A.25 A destra è schematizzata la funzione del segnale $\overline{\text{OE}}$ nella struttura a bus.

L (Basso). Nel primo caso si dice che il segnale è *attivo alto*, nel secondo si dice che è *attivo basso*. Nel tracciare gli schemi delle reti si usa la convenzione di marcare come negati i segnali che sono attivi bassi e, quando questi sono segnali di ingresso, si aggiunge un pallino di negazione al punto di ingresso del segnale stesso, come nelle Figure A.24 e A.25, dove aver negato il segnale OE sta a indicare che esso è attivo, ovvero svolge la sua funzione di abilitatore dell'uscita, quando è Basso.

Quando un segnale di controllo ha un differente significato a seconda del fatto che sia in stato H o L, si segue la convenzione di marcare il segnale con un simbolo composto dalla combinazione di due nomignoli che richiamano la funzione svolta, marcando come negato il nomignolo corrispondente alla funzione svolta in stato di basso. Le precedenti convenzioni sono illustrate in Figura A.26.

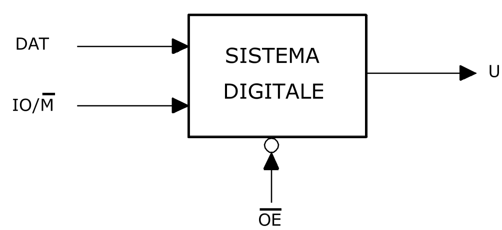


Figura A.26 Convenzioni per la marcatura dei segnali. Il segnale marcato $\overline{\text{OE}}$ ha la funzione di *Output Enable* quando è in stato L. Si dice che il segnale è asserito quando si trova in questo stato. il segnale $\text{IO}/\overline{\text{M}}$ ha significato IO (presumibilmente *Input/Output*) in stato H e significato M (presumibilmente *Memoria*) in stato L.

Sebbene queste convenzioni eliminino ogni possibile equivoco legato al tipo di logica, quando ci si riferisce ai segnali di controllo che svolgono la loro funzione in uno solo dei due stati, si usa, per brevità, la dizione di segnale *asserito* o *disasserito* a seconda che esso si trovi o no nello stato corrispondente alla funzione, evitando richiamare espressamente il fatto che questo sia alto o basso.

Purtroppo questo genere di notazione viene spesso usato in modo non congruente nei manuali dei produttori di integrati. Il miglior modo per evitare errori nell'interpretare lo schema resta quello di leggere la descrizione a parole del funzionamento del dispositivo ed esaminare i relativi diagrammi temporali.

A.6 Moduli combinatori di interesse

L'algebra di commutazione permette di analizzare e progettare qualunque tipo di rete combinatoria. Sebbene in linea di principio l'algebra può essere utilizzata per reti combinatorie di qualunque estensione, nella pratica si preferisce affrontare il progetto di una rete complessa scomponendola in sottoreti più facilmente trattabili. In questo modo si rinuncia alla soluzione teoricamente ottima a favore di una maggior comprensibilità e gestibilità del progetto. Del resto, vengono prodotti componenti integrati che svolgono precise funzionalità di carattere combinatorio e la loro utilizzazione favorisce lo sviluppo del progetti ordinati. Nella parte che segue vengono esaminati alcuni dei moduli combinatori di uso ricorrente.

A.6.1 Decodificatori

Un decodificatore (1 su m) accetta in ingresso un codice di n bit e presenta in uscita $m = 2^n$ linee, sulle quali asserisce solo quella che corrisponde alla codifica in ingresso. Numerando le linee di uscita da 0 a $2^n - 1$, viene asserita quella che corrisponde al numero presente in ingresso. Essa è l'uscita dell'AND che rappresenta il corrispondente mintermine (Paragrafo A.2.1). Un esempio di decodificatore è in Figura A.27.

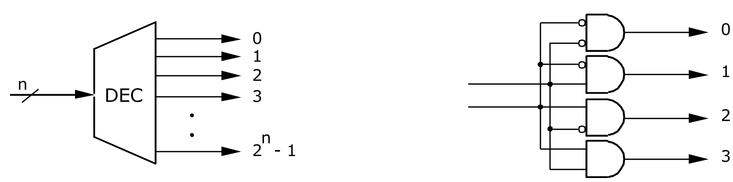


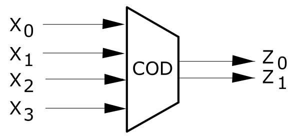
Figura A.27 Schematizzazione di un decodificatore 1 su m ($m = 2^n$) ed esempio di decodificatore 1 su 4.

A.6.2 Codificatori

Un codificatore svolge la funzione inversa di un decodificatore, nel senso che esso prevede 2^n ingressi e n uscite. Le uniche configurazioni ammesse per gli ingressi contengono esattamente un solo 1. Ovvero, indicando con $x_0, x_1, x_{2^n} - 1$ gli ingressi, con $x_i = 1$ deve essere $x_j = 0$ per ogni $j \neq i$, mentre la corrispondente configurazione di uscita su $z_0, z_1, ..., z_{n-1}$ vale i. Lo schema del codificatore è riportato in Figura A.28, assieme alla relativa tabella di verità. Si noti che la tabella di verità è stata riportata solo per le configurazioni di ingresso definite, le altre danno luogo a condizioni non specificate (di indifferenza). Nel caso di Figura A.28 le funzioni di uscita sono immediatamente esprimibili nel modo seguente[10].

$$z_0 = x_1 + x_3 \qquad\qquad z_1 = x_2 + x_3$$

[10] La presenza di molte condizioni di indifferenza sulla tabella favorisce la semplificazione delle funzioni di uscita. Nel caso specifico, se si tiene conto delle condizioni di indifferenza si ottiene $z_0 = \overline{x}_2$ e $z_1 = \overline{x}_1$.



x_3	x_2	x_1	x_0	z_1	z_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Figura A.28 Schematizzazione di un codificatore a 4 ingressi e (parte della) tabella di verità delle funzioni di uscita.

A.6.3 Selettori

Un selettore di ingresso (o *multiplexer*) è un dispositivo che permette di selezionare uno degli N ingressi e presentarlo sull'unica uscita. Un selettore di uscita (o *demultiplexer*) è un dispositivo che permette di dirottare l'ingresso su una delle possibili N uscite. La selezione si effettua attraverso linee di comando. Con N linee di ingresso un multiplexer richiede un numero di linee di comando della selezione pari all'intero uguale o superiore a $n = log_2 N$.

In Figura A.29 viene data la schematizzazione usuale dei selettori e la loro realizzazione, con riferimento a selettori a due vie. Si noti che nel caso di selettore di uscita, l'ingresso viene presentato sulla via selezionata, mentre sull'altra via viene presentata una configurazione di tutti zeri. Spesso i selettori hanno l'uscita in terzo stato (e sono quindi dotate del relativo ingresso di abilitazione).

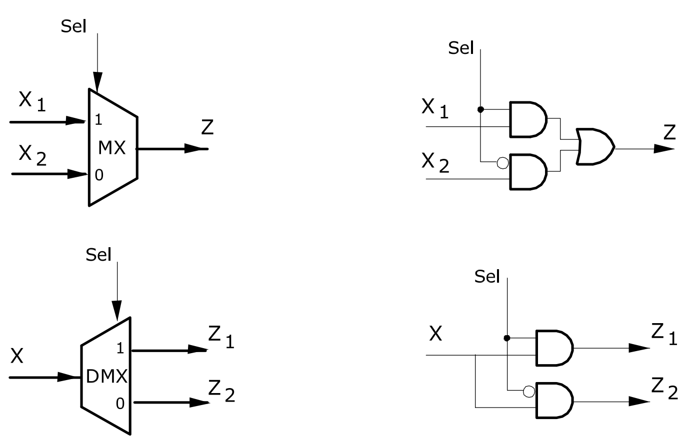


Figura A.29 Schematizzazione di un selettore di ingresso e di un selettore di uscita ambedue a due vie. A destra viene riportata la loro realizzazione. Per il selettore di ingresso il valore della linea (di controllo) Sel seleziona quale, tra x1 e x2, trasferire su z. Per il selettore di uscita la linea Sel stabilisce su quale, tra z1 e z2, deve essere presentato l'ingresso x. Ovviamente, se le vie sono più di due, il selettore Sel sarà formato da tante linee quante servono a codificarle. Inoltre, il parallelismo delle vie (dati) di ingresso/uscita può essere superiore a uno.

Vogliamo ora illustrare un impiego particolarmente interessante dei selettori di ingresso:

la costruzione di qualunque funzione logica di n variabili attraverso un selettore a $N = 2^n$ vie.

Una qualunque funzione di n variabili può essere ricondotta a una somma di mintermini (Paragrafo A.2.1). La funzione di Figura A.5 è stata scritta in forma sintetica come $y = \sum_3(2, 4, 5, 6, 7)$. Poteva essere scritta simbolicamente come

$$y = \sum_3 (0 \cdot 1,\ 0 \cdot 1,\ 1 \cdot 2,\ 0 \cdot 3,\ 1 \cdot 4,\ 1 \cdot 5,\ 1 \cdot 6,\ 1 \cdot 7)$$

ovvero in una forma in cui i mintermini sono moltiplicati per 1 e le altre combinazioni delle variabili per 0. Più sinteticamente, la forma del tutto generale di una funzione è quindi espressa

$$y = \sum_{\forall i} (c_i \cdot C_i)$$

dove c_i è un coefficiente che vale 0 o 1 e C_i è una generica combinazione delle variabili della funzione.

La funzione di Figura A.5 (della quale è stata data la forma minima A.2) scritta per esteso è:

$$\begin{aligned} y &= 1 \cdot \overline{x}_1\overline{x}_2\overline{x}_3 + 0 \cdot \overline{x}_1\overline{x}_2 x_3 + 0 \cdot \overline{x}_1 x_2 \overline{x}_3 + 1 \cdot \overline{x}_1 x_2 x_3 + \\ &\quad +0 \cdot x_1\overline{x}_2\overline{x}_3 + 0 \cdot \overline{x}_1 x_2 \overline{x}_3 + 1 \cdot x_1 x_2 \overline{x}_3 + 1 \cdot x_1 x_2 x_3 = \\ &= \sum_3 (0, 3, 6, 7) \end{aligned}$$

Con un selettore a 8 vie[11] una funzione y di tre variabili x_1, x_2, x_3 si ottiene impiegando le variabili come linee di selezione e ponendo le 8 linee di ingresso a 0 o a 1 a seconda del corrispondente c_i, come illustrato in Figura A.30.

Poiché, come abbiamo visto, ogni funzione logica può essere ricondotta alla sua prima forma canonica (Paragrafo A.2.1), ne deriva che è possibile realizzare qualunque funzione col metodo sopra esposto.

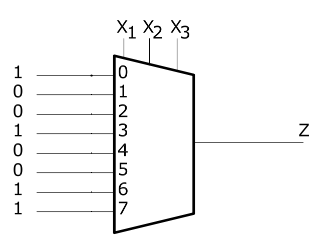


Figura A.30 Realizzazione della funzione $y = \sum_3(0, 3, 6, 7)$ tramite multiplexer a 8 vie.

È ragionevole domandarsi perché usare tale metodo. La risposta sta nel fatto che con i circuiti integrati la soluzione con multiplexer può portare a una riduzione del numero di componenti da impiegare. Ad esempio, la funzione $f = ab + b\overline{c}$ richiede due integrati (uno

[11] L'integrato SN74151 è un selettore di ingresso a 8 vie.

per gli AND e uno per l'OR). È vero che usando il multiplexer la rete effettiva è molto più complessa, ma il progettista ha come obiettivo la minimizzazione del circuito risultante, dunque l'impiego del solo multiplexer può risultare vantaggioso sotto il profilo del costo e dell'occupazione di superficie sulla piastra. Inoltre l'impiego del multiplexer dà maggior flessibilità al progetto: gli ingressi corrispondenti ai coefficienti possono essere riguardati come *ingressi di programmazione*; su una piastra elettronica possono essere aggiustati anche attraverso collegamenti ad hoc (verso Vcc o verso massa).

Al di là di queste considerazioni, che, a causa delle tecnologie correnti e dei livelli di integrazione attuali, hanno in parte perso di importanza, la realizzazione delle funzioni logiche tramite multiplexer è la tecnica standard usata nei dispositivi FPGA di cui si parla al Paragrafo A.6.6.

A.6.4 Arbitro di priorità

Spesso si presenta questo problema: dati n ingressi, ciascuno con un differente livello di priorità compreso tra 0 e $n-1$, costruire la rete che ha in uscita n linee, di cui risulta asserita solo quella corrispondente alla linea di ingresso di maggior priorità tra quelle asserite. In Figura A.31 viene schematizzato un arbitro di priorità tra quattro ingressi A, B, C, D. Assumendo che l'ingresso A sia quello più prioritario e che l'ingresso D sia

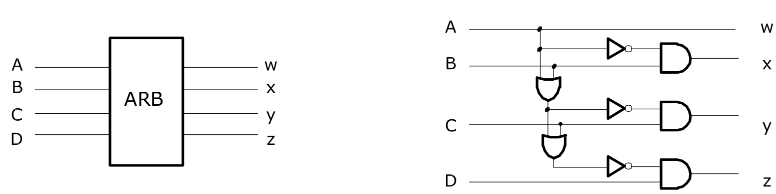


Figura A.31 Schematizzazione di un arbitro di priorità a 4 ingressi e corrispondente rete "a scala".

quello meno prioritario, si avrebbe: $w = A$; $x = B\overline{A}$; $y = C\overline{A}\ \overline{B}$; $z = D\overline{A}\ \overline{B}\ \overline{C}$.

Usando questo criterio, occorrono porte AND con crescente numero di ingressi. Se la rete ha n ingressi, l'uscita meno prioritaria richiederebbe una porta AND da n ingressi. Per questo motivo si usa una soluzione come quella a destra in Figura A.31, con la quale le porte sono sempre a due ingressi. Ovviamente questa rete richiede un maggior tempo di commutazione, a causa della propagazione dei segnali su più livelli di porte.

In Figura A.32 viene mostrato come si possa costruire un codificatore di priorità (*priority encoder*) impiegando il precedente arbitro e un modulo codificatore.

A.6.5 Memorie ROM

Riprendiamo in considerazione la prima forma canonica (Paragrafo A.2). La realizzazione di qualunque funzione di m variabili richiede un numero di porte AND a m ingressi pari al numero di mintermini e una porta OR con un numero di ingressi pari al numero di porte AND. Si tratta di "prolungare" le uscite delle porte AND agli ingressi della porta OR. In Figura A.33 viene mostrata una struttura regolare con la quale i collegamenti tra le uscite delle porte AND, cioè i mintermini, e gli ingressi della porta OR sono ottenuti attraverso contatti.

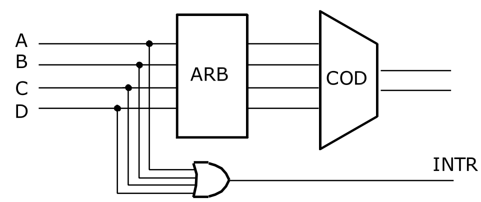


Figura A.32 Costruzione di un codificatore di priorità a partire da un arbitro di priorità e da un codificatore. La linea INTR indica che almeno una delle linee tra A, B, C e D è 1. Quando INTR è 0, nessuna linea di ingresso risulta asserita e l'uscita del codificatore è senza significato.

Fissare i contatti equivale a *programmare* il comportamento della rete (più avanti si accenna alle reali modalità di programmazione nella pratica corrente).

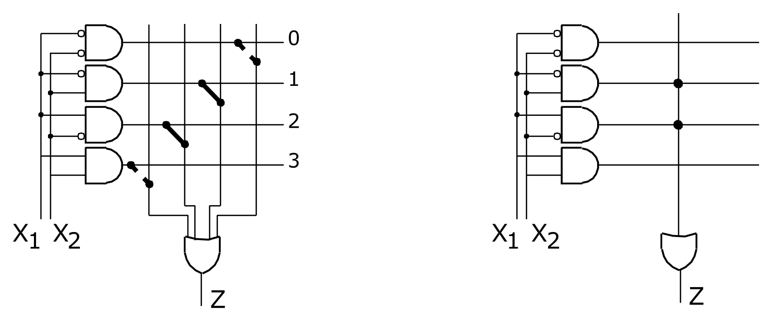


Figura A.33 Matrice di contatti che consente di ottenere qualunque funzione logica dei due variabili. L'insieme delle uscite dalle porte AND rappresenta tutti i prodotti fondamentali delle variabili di ingresso. Nel caso specifico la rete ha come uscita $z = \overline{x}_1 x_2 + x_1 \overline{x}_2 = x_1 \oplus x_2$. A destra viene riportato lo schema sintetico che rappresenta la rete; i pallini indicano gli incroci in cui si ha contatto. Lo schema a sinistra ha valore di principio in quanto la porta OR avrebbe due ingressi "volanti", cosa sconsigliata perché non è definito il comportamento della porta stessa. Nella pratica la rete deve essere costruita in modo da dare continuità elettrica, verso massa o verso l'alimentazione, di ogni suo ingresso.

La struttura di Figura A.33 può, ovviamente, essere estesa in modo da fornire non una ma più funzioni di uscita, passando, ad esempio, ad una rete con m ingressi e k uscite. Una simile rete fornisce per ogni configurazione degli m ingressi una configurazione sulle k uscite e viene a costituire una ROM (*Read Only Memory*, memoria di sola lettura), di $M = 2^m$ posizioni (celle) di k bit ciascuna[12]. In Figura A.34 viene dato lo schema a blocchi di una simile memoria. Il numero binario corrispondente alla combinazione degli m ingressi rappresenta l'indirizzo della corrispondente cella.

[12]Si chiama di sola lettura perché nel normale funzionamento può essere solo letta, la scrittura viene effettuata una volta per tutte in fase di programmazione.

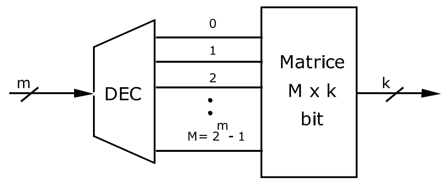


Figura A.34 Schematizzazione funzionale di una memoria ROM.

Lo schema di una ROM da 8 posizioni di 4 bit è in Figura A.35. Nella figura sono stati riportati solo i collegamenti esistenti attraverso diodi. In realtà i diodi sono previsti in ogni incrocio. Il modo in cui vengono resi attivi distingue i differenti tipi di memorie di sola lettura.

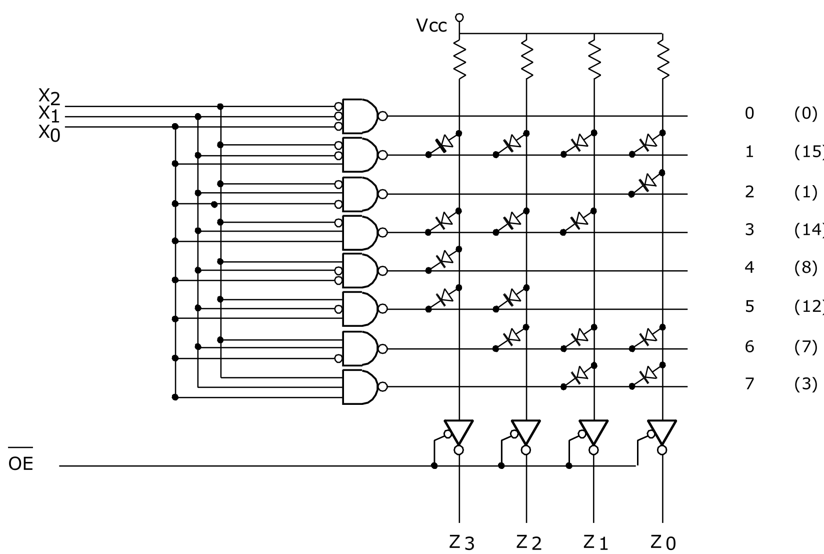


Figura A.35 Schema di una ROM di 8 posizioni di 4 bit. A destra viene riportato l'indirizzo di ciascuna cella e, tra parentesi, il suo contenuto. Per capirne il funzionamento si consideri ad esempio il caso in cui l'indirizzo $x_2x_1x_0$ è 2. In tal caso solo la riga 2 della matrice è asserita bassa, per cui la tensione sull'anodo del diodo ad essa collegato (colonna z_0) risulta bassa (il diodo ha una caduta trascurabile). Gli altri diodi sulla colonna z_0 risultano interdetti. I diodi sulle restanti colonne sono in conduzione, ma la tensione sui loro anodi è alta essendo alta la tensione su tutte le righe diverse dalla 2. Si noti che la memoria ha un segnale di abilitazione delle uscite.

Nelle ROM propriamente dette, la matrice non ha inizialmente alcun punto di contatto tra righe e colonne, come schematizzato a sinistra in Figura A.36. È il costruttore che fa le connessioni in base alla matrice di bit desiderata dal committente. Non è possibile variare il contenuto della memoria dopo la programmazione. Risultano economicamente convenienti per volumi molto grandi.



Figura A.36 Schematizzazione della modalità di interconnessione tra righe e colonne per le ROM (a sinistra) e per le PROM (a destra).

Nelle PROM (*Programmable Read Only Memory*), la matrice ha inizialmente tutti i punti di contatto tra righe e colonne, attraverso un diodo e un fusibile, come schematizzato a destra in Figura A.36. Ciò vuol dire che inizialmente tutti i bit sono a 1. Programmare un bit a 0 richiede la fusione del relativo fusibile. Tocca all'utente inserire gli 0 negli incroci desiderati, attraverso un apparato di programmazione. Non è possibile variare il contenuto della memoria dopo la programmazione. Risultano economicamente convenienti per volumi medio/grandi.

Nelle EPROM (E sta per *Erasable*), il collegamento tra righe e colonne è ottenuto con altro metodo. La memoria è in un contenitore che presenta una piastrina di quarzo attraverso la quale possono passare raggi ultravioletti che ripristinano la programmabilità (cancellando la programmazione corrente). Risultano convenienti per prototipi di laboratorio o per bassissimi volumi di produzione.

Le EEPROM (la doppia E sta per *Electrically Erasable*) consentono la loro riprogrammazione per via elettrica. Dunque sono indicate per un impiego che preveda la possibilità di modificarne il contenuto direttamente dall'apparato in cui vengono usate.

A.6.6 Matrici di logica programmabili

Una PROM è un dispositivo programmabile dall'utente[13] in cui è presente una matrice di AND fissa e una matrice di OR programmabile. La matrice di AND è fissata una volta per tutte dal costruttore, la matrice di OR può essere programmata dal progettista per realizzare le funzion i desiderate. La Figura A.37 schematizza tale organizzazione.

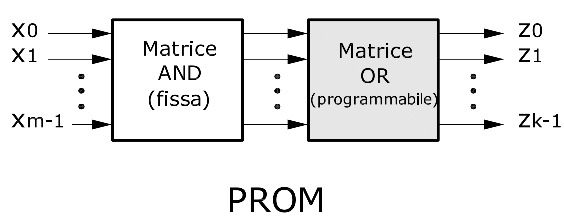


Figura A.37 Organizzazione di una PROM.

La programmazione consiste nel fissare quali ingressi vengono portati alle porte OR. In Figura A.38 viene data la rappresentazione di una PROM corrispondente a quella del-

[13] In generale si parla di dispositivi programmabili in campo (*Field Programmable Device*).

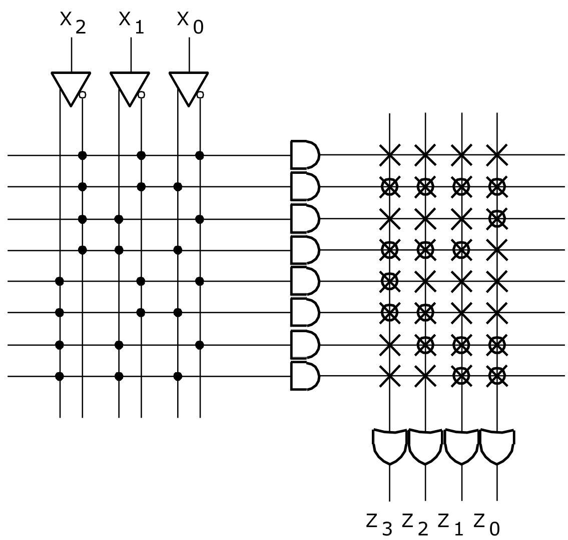


Figura A.38 Schema di una PROM 8 per 4. Gli incroci della matrice programmabile sono stati indicati con "×". Quelli in cui è tracciato anche un cerchio sono quelli in cui c'è continuità elettrica, ovvero contengono un 1. La programmazione corrisponde a quella di Figura A.35. Si noti il nuovo simbolo usato per la rappresentazione di un segnale e del suo negato.

la ROM 8×4 di Figura A.35. La figura è stata tracciata ricorrendo alla rappresentazione schematica delle connessioni introdotta in Figura A.33. Sugli incroci della matrice programmabile sono stati posti degli "×". Sugli incroci programmati a 1 (dove si stabilisce continuità elettrica) oltre al simbolo "×" è stata riportata una cerchiatura.

Vi sono altre forme di dispositivi a matrici programmabili, con denominazioni capaci di generare qualche confusione.

PAL (*Programmable Array Logic*)

Questi dispositivi hanno una matrice di OR fissa, con connessioni stabilite dal costruttore, e una matrice di AND programmabile, come illustrato a sinistra in Figura A.39. In sostanza lo schema è il duale di quello delle PROM (Figura A.37).

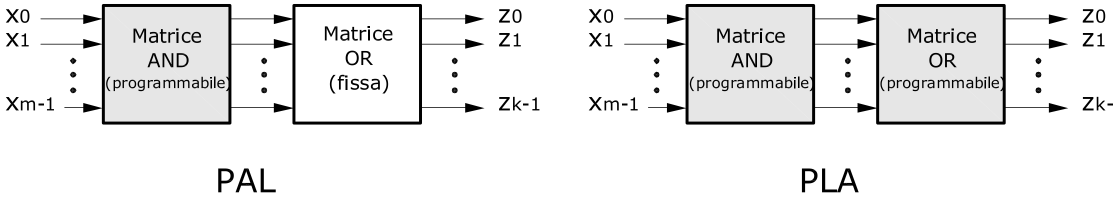


Figura A.39 Differenti organizzazioni di ROM, PAL e PLA.

PLA (*Programmable Logic Array*)

Si tratta di dispositivi in cui è programmabile sia la matrice di AND sia la matrice di OR, come illustrato a destra in Figura A.39; mentre in Figura A.40 viene data la schematizzazione di un dispositivo a 4 ingressi e due uscite. Le PLA consentono il massimo della flessibilità.

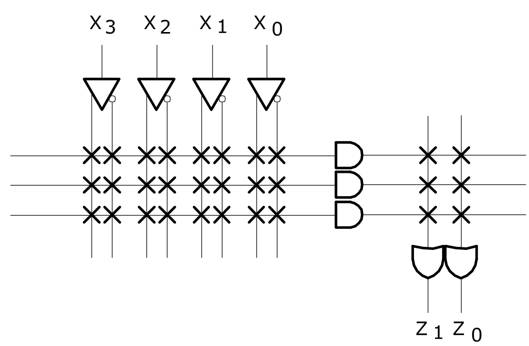


Figura A.40 Schema di un dispositivo PLA a quattro ingressi e due uscite. Il dispositivo prevede una matrice di AND e una matrice di OR, ambedue programmabili in campo.

Per tutti i dispositivi a matrici logiche programmabili visti in precedenza, la programmazione consiste nel definire le interconnessioni necessarie all'implementazione delle funzioni logiche desiderate sulla matrice di interconnessione predefinita. Ovviamente la programmazione e la trasformazione dell'integrato vergine in un componente con definite caratteristiche funzionali si attua attraverso opportuni strumenti software e hardware di supporto.

I dispositivi menzionati servono alla creazione di logica combinatoria. Per la creazione di logica più complessa occorre utilizzare dispositivi come quelli che descriviamo brevemente qui di seguito.

Gate Array

I Gate Array tradizionali, ovvero i cosiddetti MPGA (*Mask Programmable Gate Array*), sono costituiti da matrici di transistori che, all'origine, non sono interconnessi tra loro. L'interconnessione avviene a seguito del progetto logico, necessariamente svolto con appropriati strumenti CAD che mappano la logica del progetto (porte, flip-flop, ecc.) sui transistori contenuti nel chip, definendo le interconnessioni che servono a dar vita alla logica specificata. I CAD in questione sono equipaggiati di librerie che definiscono *macrocelle* e *macrofunzioni* (per esempio, decodificatori, multiplexer ecc.) utilizzabili nel progetto, in modo da liberare il progettista dal dover definire in dettaglio parti ricorrenti. Inoltre essi permettono la simulazione e la validazione del progetto.

I Gate Array sono tipicamente impiegati nella realizzazione di circuiti ASIC (*Application Specific Integrated Circuit*), cioè circuiti sviluppati per una funzione ben definita. Lo sviluppo di dispositivi ASIC richiede di solito tempo e danaro, ed è giustificato quando c'è la prospettiva di alti numeri di produzione e/o vendita.

FPGA (*Field Programmable Gate Array*)

I dispositivi FPGA sono stati introdotti a metà degli anni ottanta e oggi godono di

grande popolarità. Essi possono essere riguardati come l'evoluzione dei precedenti. Il termine FPGA deriva dal fatto che la sua architettura è simile a quella dei gate array, ma con la differenza che essi possono essere programmati dall'utente, con apparati di costo minimo.

Lo schema di un FPGA è mostrato in Figura A.41. Mentre nei gate array la matrice è costituita da transistori, negli FPGA gli elementi della matrice sono blocchi di logica configurabile (CLB), detti anche celle logiche, i quali possono contenere multiplexer, flip-flop e memorie. Il sistema di interconnessione è composto da linee di collegamento e da commutatori (switch) programmabili, realizzati tramite transistori. I blocchi di I/O fanno da interfaccia con il mondo esterno. La presenza dei commutatori ha l'effetto negativo di far aumentare i tempi di propagazione dei segnali; tipicamente un dispositivo realizzato con FPGA è mediamente tre volte più lento di un corrispondente realizzato con gate array [BFRV92].

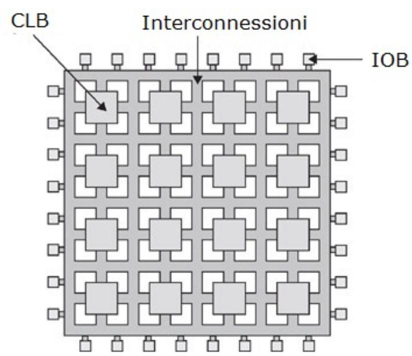


Figura A.41 Schematizzazione di un FPGA. CLB sta per *Configurable Logic Block*, IOB per *I/O Block*

I blocchi di logica configurabile sono gli elementi distintivi dei FPGA. Essi contengono tre componenti fondamentali: flip-flop, multiplexer e le cosiddette LUT (*Look Up Table*). Lo schema di un semplice CLB è in Figura A.42.

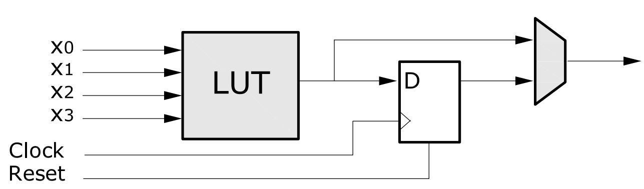


Figura A.42 Esempio di blocco di logica programmabile (CLB). Le parti in grigio sono quelle programmabili.

Una LUT è programmabile e può realizzare qualunque funzione delle variabili di ingresso (di norma 4 o 6), secondo lo schema descritto al Paragrafo A.6.3, Figura A.30. A tale scopo la LUT contiene una memoria RAM (statica) che viene caricata al momento

della messa sotto tensione del dispositivo. Dare un contenuto della RAM corrisponde ad assegnare i valori (1/0) a quelli che in Figura A.30 sono gli ingressi al selettore. Analogamente, è possibile programmare il selettore finale; nel caso di figura, a presentare o la funzione generata dalla LUT o lo stato del flip-flop.

Dalla descrizione precedente si evince che un dispositivo FPGA non è in grado di operare fino a che non è stata caricata la configurazione. Di norma questa è tenuta in una memoria flash esterna e da qui trasferita all'FPGA al momento della messa sotto tensione. Alcuni dispositivi prevedono direttamente al loro interno una copia, o addirittura due copie, dei dati di inizializzazione del dispositivo.

Come per gate array, per la progettazione si ricorre a linguaggi HDL (*Hardware Description Language*), come ad esempio VHDL e Verilog. Questi linguaggi servono a definire i blocchi di hardware che compaiono nel progetto. Diversamente dai tradizionali linguaggi di programmazione, essi non danno luogo ad alcuna esecuzione sequenziale. La descrizione HDL viene trasformata in una *netlist*, una lista delle connessioni; successivamente, un sintetizzatore, basato sulle librerie di mappatura fornite dal produttore di FPGA, mappa le funzionalità descritte sull'hardware reale.

Il grande vantaggio dei FPGA è la loro flessibilità. La produzione di un dispositivo finito non richiede grandi investimenti, può essere portata a termine in tempi relativamente brevi, con costi nettamente inferiori a quelli richiesti per lo sviluppo di dispositivi ASIC. A confronto, questi ultimi sono assolutamente rigidi, mentre il comportamento degli FPGA può essere aggiornato nel tempo. Da questo punto di vista essi sono il mezzo ideale per fasi di prototipizzazione o di sperimentazione. Anche i produttori di componenti complessi, come i processori, possono trovare negli FPGA il modo più idoneo per sperimentare le loro soluzioni, prima di passare all'implementazione definitiva.

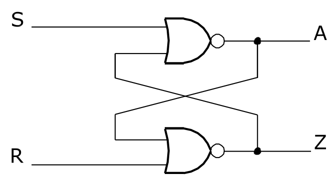
Gli FPGA sono evoluti fino al punto di prevedere al loro interno elementi predefiniti, attorno ai quali può essere costruita la logica di supporto che conduce alla realizzazione di completi sistemi sul chip (SoC). I principali produttori di FPGA (e.g., Xilink, Altera) hanno a listino dispositivi che al loro interno contengono preconfezionato un processore. Si tratta quasi sempre di una qualche versione di processore ARM-Cortex, a due o più core. Il risultato è un dispositivo SoC che permette di integrare la programmabilità software del processore con la programmabilità hardware dell'FPGA.

A.7 Reti sequenziali

Le reti esaminate in precedenza erano tutte di tipo combinatorio. L'uscita di una rete combinatoria è funzione del solo ingresso. Quando l'uscita è funzione anche dello *stato* si parla di reti sequenziali. Avviciniamoci alle reti sequenziali analizzando il comportamento della rete di Figura A.43, detta *latch di NOR*.

Per la configurazione di ingresso 00 in uscita sono possibili due configurazioni: 01 e 10 (per questo nella tabella l'ingresso 00 è stato riportato due volte). Per convincersi di ciò, è sufficiente ricordare che una porta NOR a due ingressi si comporta come un negatore del secondo ingresso se il primo è a 0.

Se si esclude il caso $(S = 1, R = 1)$, l' uscita Z risulta essere sempre uguale al complemento dell'uscita A. Per ragioni chiarite in seguito, conviene stabilire che alla rete di Figura A.43 non venga mai presentato l'ingresso $(S = 1, R = 1)$[14]. Si ha così il latch di NAND, per il quale valgono proprietà duali rispetto a quelle descritte. In particolare, l'ingresso escluso è $(S = 0, R = 0)$.



S	R	Z	A
0	0	0	1
0	0	1	0
0	1	0	1
1	0	1	0
1	1	0	0

Figura A.43 Il Latch di NOR e relative relazioni di ingresso/uscita. Il termine anglosassone *latch* significa lucchetto, chiavistello ed esprime, in modo figurato, il comportamento della rete.

Per dire quanto valgono le uscite in corrispondenza dell'ingresso 00, occorre esaminare nel dettaglio il comportamento della rete, tenendo conto del fatto che le porte reali hanno un proprio tempo di commutazione. Un porta reale può essere modellata come una porta ideale, a tempo di commutazione nullo, seguita da un elemento di ritardo che ne rappresenta il tempo di commutazione τ. Si consideri dunque la rete di Figura A.44, che risulta dall'uso di un tale modello. Si ipotizzi che ambedue gli ingressi siano 0 e si assuma la condizione iniziale $(Z = 1,\ A = 0)$. È ovvio che, se S ed R restano a 0, A e Z non cambiano.

Si supponga ora che, all'istante t, l'ingresso R si porti a 1. Istantaneamente l'uscita Z' della porta ideale P2 si porta a 0, per cui, dall'istante t fino all'istante $t+\tau$, si verifica la condizione $Z' \neq Z$. Si dice che la rete si trova in uno *stato instabile*. All'istante $t + \tau$ anche Z passa a 0 e quindi la porta di sinistra (P1), cui sono applicati due zeri, commuta la sua uscita A' da 0 a 1. Dall'istante $t+\tau$ all'istante $t+2\tau$, la rete si trova ancora in uno stato instabile, con A' diverso da A. Alla fine anche A si porta ad 1. È facile convincersi che da questo momento, per tutto il tempo in cui si mantiene l'ingresso $(S = 0, R = 1)$, l'uscita non varia; resta cioè $(Z = 0, A = 1)$. Si dice che la rete si è portata in uno *stato stabile*.

[14] Il latch, può essere realizzato impiegando una coppia di NAND al posto della coppia di NOR (si veda l'Esercizio A.27).

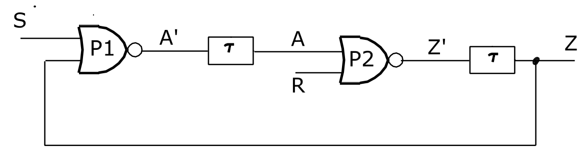


Figura A.44 Il latch di NOR ridisegnato con gli elementi di ritardo. P1 e P2 sono porte ideali e commutano in tempo nullo.

A partire dallo stato stabile raggiunto ($Z = 0, A = 1$) si riporti R a 0. È facile verificare che niente accade in uscita alla porta P1 di Figura A.44 e che la rete si mantiene nello stato precedente, e cioè con ($Z = 0, A = 1$).

A partire da questo stato si passi da ($S = 0, R = 0$) ad ($S = 1, R = 0$). Immediatamente la porta ideale P1 commuta ed A' si porta a 0. La rete entra in uno stato di instabilità con A' diverso da A e vi rimane per un tempo τ, scaduto il quale la porta P2 commuta, portando Z' ad 1 e dando luogo alla condizione di instabilità $Z' \neq Z$ che, a sua volta, ha durata pari a τ. Trascorso il tempo 2τ dal momento del passaggio ad 1 di S, anche Z si porta ad 1 e la rete è di nuovo in uno stato stabile. Se, a partire da questo stato stabile, S torna a 0, l'uscita non si modifica.

Si sottolinea che nei precedenti ragionamenti si è implicitamente assunto che gli ingressi cambino solo mentre la rete è in uno stato stabile. In altri termini, si è assunto che se si ha un cambiamento degli ingressi dal quale può derivare un cambiamento di stato, gli ingressi non cambiano ulteriormente almeno fino a che la rete non è pervenuta al prossimo stato stabile.

È possibile trarre alcune conclusioni dai ragionamenti precedenti. Prima, però, conviene fare un'altra piccola astrazione e sostituire al modello di Figura A.44 quello più conveniente, ma perfettamente equivalente, di Figura A.45, che concentra in un solo punto il ritardo, assunto per comodità ancora pari a τ

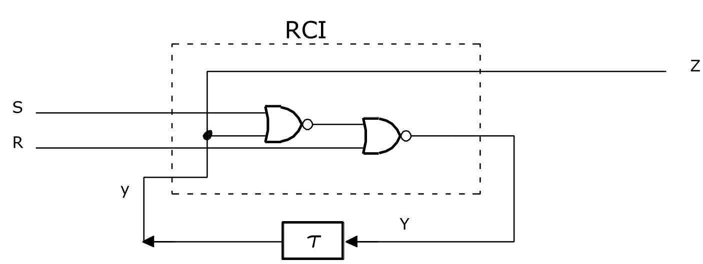


Figura A.45 Modello di latch con un solo ritardo concentrato τ. RCI è una rete combinatoria ideale con tempo di commutazione nullo.

Sullo schema di Figura A.45 sono stati riportati i simboli Y e y. La variabile y rappresenta lo stato presente, la variabile Y rappresenta invece lo stato futuro. La condizione di stato stabile corrisponde alla situazione $y = Y$, mentre la condizione di stato instabile corrisponde alla situazione $y \neq Y$. Quando la rete entra in una condizione di stato instabile, dopo il tempo τ, y assume il valore a cui si è portato Y. Nel caso specifico del

latch di NOR, il nuovo valore di y presentato a RCI fa generare $Y = y$. In altri termini la rete ha raggiunto il prossimo stato stabile.

Si noti che, con $(S = 0, R = 0)$, l'uscita Z dice quale tra S e R ha assunto per ultimo il valore 1: se $Z = 1$, la precedente configurazione di ingresso è stata $(S = 1, R = 0)$. In altri termini, il latch di NOR ha memoria di quale dei due ingressi è passato per ultimo a 1[15]. Un dispositivo come questo costituisce un elemento binario di memoria, detto anche *flip-flop*. Nel caso specifico si parla di flip-flop *Set-Reset* (asincrono). Esso verrà indicato come FFSR. La sua rappresentazione convenzionale è in Figura A.46a.

Riferendosi alla Figura A.45, lo stato futuro Y può essere espresso in funzione degli ingressi S ed R, e dello stato presente y. In Figura A.46 sono riportate la tabella di verità e la corrispondente mappa di Karnaugh. Da queste si deduce l'equazione di stato del flip-flop SR:

$$Y = S + \overline{R}y \qquad \text{con il vincolo} \quad SR = 0 \tag{A.4}$$

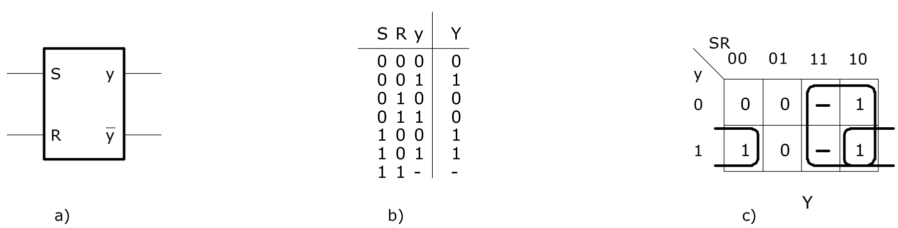


Figura A.46 Il flip-flop SR asincrono: a) schema convenzionale; b) tabella di verità; c) mappa di Karnaugh dello stato futuro.

A.7.1 Modello generale di rete sequenziale

In Figura A.47 viene riportato il modello del tutto generale di una rete sequenziale ottenuto estendendo quello di Figura A.45. Si definisce:

– il vettore delle variabili di ingresso: $X = (x_1, x_2, ..., x_n)$
– il vettore delle variabili di uscita: $Z = (z_1, z_2, ..., z_m)$
– il vettore delle variabili di stato presente: $y = (y_1, y_2, ..., y_l)$
– il vettore delle variabili di stato futuro: $Y = (Y_1, Y_2, ..., Y_l)$

Ovviamente x_i, y_i, z_i, Y_i sono definite su {0, 1}.

Una n-pla $(x_1, x_2, ..., x_n)$ costituisce una configurazione d'ingresso o, più semplicemente, un ingresso. L' insieme delle $N = 2^n$ configurazioni d'ingresso costituisce l'alfabeto d'ingresso $I = \{I_1, I_2, ..., I_N\}$; ogni configurazione I_k costituisce un simbolo dell'alfabeto. Analogamente si definiscono gli alfabeti di uscita e di stato:

[15] Possiamo ora spiegare perché si è stabilito che il latch non debba essere mai sottoposto all'ingresso $S = R = 1$. Innanzitutto si avrebbe $Z = A = 0$, diversamente da tutti gli altri casi per i quali vale sempre $Z = \overline{A}$. Secondariamente, se a partire da $S = R = 1$ gli ingressi tornano a zero lo stato finale è impredicibile, in quanto questo risulterebbe casualmente determinato dai tempi di commutazione delle singole porte oltre che dal grado di coincidenza dei fronti di discesa di S e R.

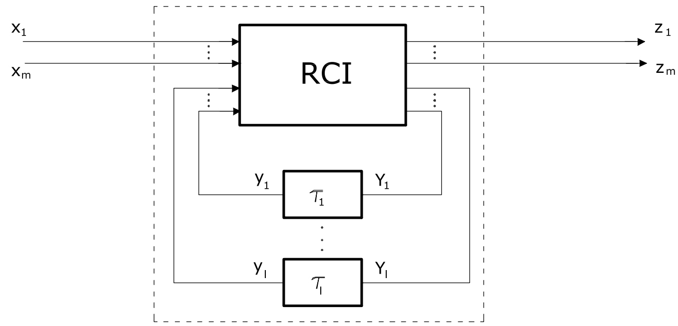


Figura A.47 Modello generale di rete sequenziale.

$$O = \{O_1, O_2, ..., O_M\} \qquad \text{con} \qquad M = 2^m$$
$$S = \{S_1, S_2, ..., S_L\} \qquad \text{con} \qquad L = 2^l$$

dove il simbolo O_i rappresenta un'uscita del sistema e il simbolo S_i rappresenta uno stato. Si definisce *macchina sequenziale* la quintupla:

$$M = (I, O, S, f, g) \tag{A.5}$$

dove I, O e S sono rispettivamente gli alfabeti d'ingresso, di uscita e di stato e f e g sono le due funzioni:

$$f : S \times I \to O \tag{A.6}$$
$$g : S \times I \to S \tag{A.7}$$

L'alfabeto di stato è finito, dunque la macchina ha una memoria finita. Di conseguenza il simbolo di uscita dipende soltanto dagli ultimi k simboli d'ingresso. Per questo si parla anche di *macchine* ovvero di *automi a stati finiti.*

Il modello precedente, nel quale l'uscita è funzione di S e di I, viene detto modello di Mealy. Quando la relazione d'uscita è del tipo: $f : S \to O$, si parla di modello di Moore.

Reti sequenziale asincrone.

Lo schema di Figura A.47 è l'interpretazione di una rete composta da porte con tempi di commutazione nulli (racchiusa in RCI) e da elementi di ritardo corrispondenti ai tempi di commutazione delle porte sugli anelli di retroazione. Il modello di Figura A.47 indica chiaramente che la rete sequenziale si trova in uno stato stabile se il vettore Y che forma parte dell'uscita della rete combinatoria RCI è uguale al vettore y in presenza del vettore complessivo d'ingresso a RCI dato da (X, y). Se il vettore d'ingresso X cambia, RCI genera, con tempo nullo, un nuovo vettore Z e un nuovo vettore Y. Se $Y \neq y$ la rete è in uno stato instabile. Senza perdere di generalità, si supponga che il ritardo sugli anelli di retroazione sia uguale per tutti, e cioè: $\tau_1 = \tau_2 = ... = \tau_l = \tau$. Dopo l'intervallo di tempo τ, y diventa uguale a Y e, conseguentemente, appare un nuovo vettore (X, y) in ingresso a RCI. Se, per effetto di questo vettore d'ingresso Y non varia, la rete ha raggiunto uno stato stabile.

La rete viene detta *asincrona* perché essa reagisce immediatamente alle variazioni dell'ingresso, portandosi nello stato (futuro) previsto dalla funzione di transizione di stato[16].

È possibile che il raggiungimento dello stato stabile richieda il passaggio attraverso una sequenza di stati instabili. È necessario assumere che, per un dato ingresso, la rete raggiunga comunque uno stato stabile[17]. Si assume inoltre che per tutto il tempo richiesto per arrivare al nuovo stato stabile il vettore di ingresso non vari. Inoltre, è necessario imporre un altro vincolo: non è permessa la variazione contemporanea di due o più variabili di ingresso. La ragione di questo vincolo è presto detta. Essendo la rete asincrona essa reagisce immediatamente alle variazioni degli ingressi. Assumere che due segnali variano contemporaneamente equivale ad assumere che tra i fronti corrispondenti alle variazioni non c'è nessun sfasamento. Questa condizione è impossibile da garantire in una rete reale, perché dipende dai percorsi che i segnali fanno. Se è previsto che due segnali varino allo stesso istante, è molto probabile che uno dei due cambi prima dell'altro: la rete asincrona reagirebbe alla sua variazione. Ciò potrebbe portare al raggiungimento di uno stato stabile in cui la sopravvenuta variazione del secondo ingresso potrebbe non aver più alcun effetto. Il fenomeno prende il nome di *corsa critica*[18].

Reti sequenziali sincrone.
Se con riferimento al modello di Figura A.47 si suppone di forzare gli elementi di memoria a cambiare stato in concomitanza di un segnale (aggiuntivo) di sicronizzazione, che chiameremo *clock*, i problemi relativi alle corse e agli stati non stabili vengono eliminati. Le reti sincrone hanno un funzionamento molto più affidabile delle asincrone. Di fatto un calcolatore è a tutti gli effetti un grosso insieme di (sotto) reti sincrone.

A.7.2 Rappresentazione delle funzioni di stato e di uscita

Ci sono due modi per descrivere il comportamento delle macchine a stati finiti: i diagrammi di stato e le tabelle di flusso. In ambedue i casi si tratta di dare una descrizione delle due funzioni f e g che definiscono l'automa e che qui vengono riscritte come:

$$O = f(I, S_p) \tag{A.8}$$

$$S_f = g(I, S_p) \tag{A.9}$$

dove la (A.8) indica che il simbolo di uscita è determinato dal simbolo d'ingresso e dallo stato (presente) della rete, mentre la (A.9) dice che, con il simbolo d'ingresso I, la rete si porterà dallo stato presente S_p allo stato futuro S_f.

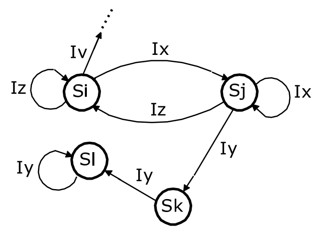
Un diagramma di stato è un grafo orientato in cui i nodi rappresentano gli stati della rete, mentre gli archi rappresentano le transizioni. Un frammento di diagramma di stato è riportato in Figura A.48a. Esso indica che se la rete è nello stato S_i, vi resta fintanto che $I = I_z$. Se a partire dallo stato S_i, l'ingresso diventa I_x la rete si porta nello stato

[16] La precedente discussione è stata riferita al modello di Figura A.47, dove c'è una rete combinatoria con tempi di commutazione nulli e dove tutti i ritardi sono concentrati sugli anelli di retroazione. È evidente che, tolti gli elementi di ritardo e riportata la rete combinatoria a una rete reale, il modello di Figura A.47 costituisce lo schema di principio di una *rete sequenziale asincrona*.

[17] In altre parole, si esclude che la rete si comporti come un *oscillatore*.

[18] Corsa perché c'è una "corsa" tra le variabili di stato a chi cambia prima; critica perché porterebbe in uno stato diverso da quello previsto dalla tabella di flusso (vedi più avanti). Si parla semplicemente di corsa (non critica) quando il risultato della corsa porta comunque nello stato previsto.

S_j e vi rimane per tutto il tempo in cui $I = I_x$. Dallo stato S_j la rete torna in S_i se l'ingresso ridiventa I_z, mentre se l'ingresso diventa I_y la rete passa attraverso uno stato instabile S_k e quindi si porta nello stato stabile S_l. Da S_i, se l'ingresso diventa I_v, la rete si porta in un qualche altro stato non indicato in figura.



a)

	I1	...	Ix	Iy	Iz	...	IN
S1							
::							
Si			Sj		(Si)		
Sj			(Sj)	Sk	Si		
Sk				Sl			
Sl				(Sl)			
SL							

b)

Figura A.48 Diagrammi di stato e tabelle di flusso: a) esempio di (parte di) diagramma di stato; b) tabella di flusso corrispondente.

Il diagramma degli stati offre una rappresentazione molto intuitiva, ma il suo contenuto non è formalmente elaborabile. Per questo motivo si ricorre alle tabelle di flusso, traduzione tabellare dei diagrammi di stato, facilmente riconducibili a funzioni booleane in forma tabellare.

Il frammento di diagramma di stato di Figura A.48a è stato riportato nella corrispondente tabella di flusso di Figura A.48b. Nelle tabelle di flusso di reti asincrone si usa cerchiare le caselle che corrispondono a condizioni di stabilità. Per esempio, lo stato S_j è stabile rispetto a I_x, mentre è instabile rispetto a I_y e I_z (si veda Figura A.48).

Sul diagramma di stato e sulla tabella di flusso si riportano anche le uscite della rete. Bisogna però distinguere tra il modello di Mealy e il modello di Moore. Nel caso di modello di Mealy la rappresentazione è quella di Figura A.49.



Figura A.49 Rappresentazione delle uscite col modello di Mealy.

L'interpretazione è che la rete in presenza dello stato S_j e dell'ingresso I_z si porta nello stato S_i e genera l'uscita O_{jz}. Vale la pena di rimarcare che O_{jz} è l'uscita che si ha nello stato S_j e con ingresso I_z, mentre S_i è lo stato in cui si troverà a transizione effettuata a partire da S_j con l'ingresso I_z.

Il caso di modello di Moore è schematizzato in Figura A.50. Il simbolo di uscita si trascrive nel cerchio accanto al nome dello stato, mentre la tabella di flusso si modifica in quanto per l'uscita basta scorporare un vettore di $L = 2^l$ posizioni.



Figura A.50 Rappresentazione delle uscite col modello di Moore.

Il diagramma di stato e la tabella di flusso sono utili sia nei problemi di analisi sia in quelli di sintesi. Nell'analisi, il diagramma di stato e la tabella di flusso, ricostruiti a partire dallo schema della rete, permettono di interpretarne il comportamento. Nella sintesi, partendo dalla tabella di flusso, si codificano gli stati e si ottiene la cosiddetta *tabella delle transizioni*, la quale rappresenta le variabili di stato futuro in funzione dello stato della rete e dell'ingresso. La realizzazione di queste funzioni e di quelle che esprimono l'uscita corrisponde alla stesura della rete. Dati N stati simbolici, occorre almeno un numero di elementi di memoria pari all'intero uguale o subito superiore a $\log_2 N$. Gli elementi di memoria, fino a questo punto sono stati modellati con ritardi, come in Figura A.47. Tra poco impareremo ad usare i flip-flop come espliciti elementi di memoria.

A.8 Sincronizzazione

Si consideri la rete di Figura A.51, nella quale al latch di NOR sono state aggiunte due porte AND e un ulteriore ingresso.

È facile convincersi che quando il segnale Ck vale zero, si ha che $S' = 0$ e $R' = 0$, indipendentemente dal valore di S e di R. In altri termini il latch non cambia stato se $Ck = 0$. Invece, quando $Ck = 1$ le porte AND sono *trasparenti* rispetto a S e R e il latch ha il comportamento descritto in precedenza. In Figura A.51 si ha un esempio di possibili andamenti temporali. Per ragioni che saranno chiarite in seguito, il segnale Ck è normalmente un segnale periodico. Anche se non è essenziale che Ck sia periodico, questa assunzione semplifica l'esposizione dei concetti che seguono. Per tale motivo esso viene denominato *clock*, a indicare che il successivo passaggio a 1 del segnale Ck corrisponde all'avanzare del tempo.

Si indichi con T il periodo di Ck, con Δ_1 la frazione di T durante la quale $Ck = 1$ e con Δ_2 la frazione di T durante la quale $Ck = 0$. Si faccia inoltre l'ipotesi che durante l'intervallo Δ_1 i segnali S e R non varino e che Δ_1 sia sufficiente a far completare l'eventuale transizione di stato del latch se questa è prevista.

La rete di Figura A.51 ha questa caratteristica: essa si accorge della presenza di un ingresso di Set ($S = 1, R = 0$) o di Reset ($S = 0, R = 1$) solo in corrispondenza degli intervalli di tempo in cui $Ck = 1$. In altre parole, l'aggiunta delle due porte AND e del segnale Ck ha permesso di sincronizzare l'attività del latch rispetto al segnale Ck.

A.8.1 I flip-flop (sincroni)

Il flip-flop SR

La rete di Figura A.51 costituisce, seppure in forma rudimentale (si veda il Paragra-

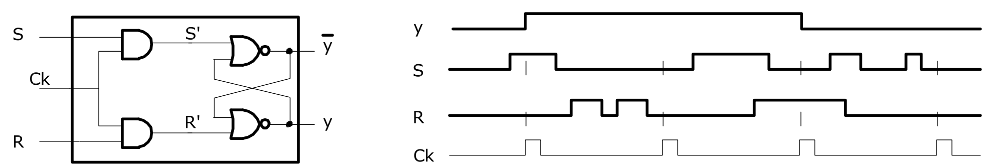


Figura A.51 Trasformazione del latch di NOR in flip-flop sincrono e possibili andamenti temporali dei segnali. Si assume che il tempo di commutazione delle porte sia nullo.

fo A.8.3), un flip-flop SR sincrono. Il FFSR sincrono viene schematizzato come in Figura A.52[19].

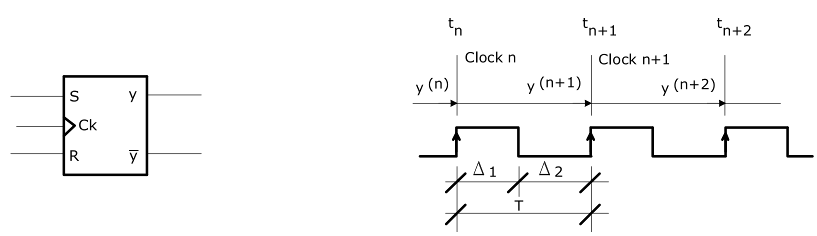


Figura A.52 Schema convenzionale del flip-flop SR sincrono e interpretazione dello stato presente e dello stato futuro rispetto a successivi impulsi di clock.

Ovviamente la funzione di transizione dello stato di FFSR sincrono resta quella di FFSR asincrono. Tuttavia è necessario mettere in evidenza il fatto che l'attività del flip-flop sincrono procede per eventi discreti. A tal fine si usa indicare con y la variabile di stato e si scrive:

$$y^{(n+1)} = S + \overline{R}y^{(n)} \tag{A.10}$$

per indicare che lo stato in cui si troverà il flip-flop sull'impulso $n+1$-mo dipende dallo stato in cui il flip-flop si trovava sull'impulso n-mo e dai valori di S e R su tale impulso.

Il concetto è sottile e merita di essere ulteriormente discusso. A tal fine si faccia riferimento alla Figura A.52. L'impulso n-mo inizia all'istante t_n; a tale istante lo stato del flip-flop (conseguente alla storia passata) è $y^{(n)}$. Durante Δ_1 a causa dell'ingresso (S, R), che per ipotesi non varia entro Δ_1, il flip-flop cambia stato e si porta in $y^{(n+1)}$. Lo stato $y^{(n+1)}$ si mantiene fino all'$(n+1)$-mo impulso di clock. Ovvero, lo stato in cui si trova il flip-flop sullo $(n+1)$-mo impulso di clock deriva dallo stato in cui si trovava sullo n-mo e dal conseguente ingresso.

Frequentemente al posto della (A.10) si usa la notazione semplificata:

$$y' = S + \overline{R}y \tag{A.11}$$

[19] D'ora in avanti, a meno di indicazione contraria esplicita, quando si parlerà dei flip-flop si intenderà parlare di flip-flop sincroni.

dove y' è lo stato in cui si trova il flip-flop, sull'impulso successivo a quello su cui si trovava in y.

Il flip-flop JK

Molto simile al flip-flop Set-Reset è il flip-flop JK (FFJK) schematizzato in Figura A.53a. Il comportamento è identico per gli ingressi 00, 01 e 10, mentre per l'ingresso 11 FF-JK cambia sempre stato. Il comportamento del FFJK è descritto dalla tabella di Figura A.53b, alla quale corrisponde la mappa di Karnaugh di Figura A.53c e da cui si deriva:

$$y' = \overline{y}J + y\overline{K} \tag{A.12}$$

In Figura A.53d viene mostrato come si ottiene un FFJK da un FFSR.

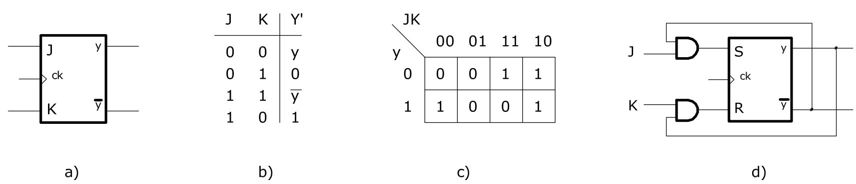


Figura A.53 Il flip-flop JK: a) simbolo schematico; b) tabella dello stato futuro; c) mappa dello stato futuro; d) JK ottenuto da SR.

Il flip-flop D

Il flip-flop D (da *Delay*) realizza un blocco di ritardo pari al periodo del clock T. L'ingresso a FFD è la materializzazione del suo stato futuro. L'equazione di transizione dello stato è banalmente

$$y' = D \tag{A.13}$$

Il simbolo del flip-flop è a sinistra in Figura A.54. Notare che y', lo stato futuro, è lo stato che si avrà nel prossimo impulso di clock in conseguenza dell'ingresso campionato dall'impulso corrente. Il FFD si ottiene facilmente da un FFSR (da un FFJK) collegando a S (J) l'ingresso D e a R (K) $\overline{D}$.

Il flip-flop T

Il flip-flop T (da *Trigger*) è schematizzato in Figura A.54. FFT cambia stato quando l'ingresso T vale 1 e resta nello stesso stato se l'ingresso è 0. Ciò comporta questa equazione di transizione di stato:

$$y' = \overline{T}y + T\overline{y} = T \oplus y \tag{A.14}$$

Il FFT si ottiene facilmente da un FFJK collegando sia a J che a K l'ingresso T.

A.8.2 Ingressi asincroni dei flip-flop.

I flip-flop, oltre agli ingressi sincroni, spesso presentano anche ingressi asincroni, detti normalmente *Preset* e *Clear*. Questi corrispondono grosso modo ai comandi di Set e Reset visti per i latch e comandano la commutazione in maniera indipendente dal clock. In presenza del segnale del clock e di segnali asincroni asseriti, di norma questi ultimi



Figura A.54 Schemi convenzionali dei flip-flop FFD e FFT. Modo in ci vengono ottenuti da FFSR/FFJK o da FFJK rispettivamente.

prevalvono. Tuttavia è bene leggere attentamente i dati tecnici di ciascun dispositivo, perché, a seconda dei modelli, è possibile anche il contrario.

I comandi PR e CL sono comodi nelle fasi di inizializzazione. Per azzerare il flip-flop all'atto della messa sotto tensione, basta un circuito come quello di Figura A.55.

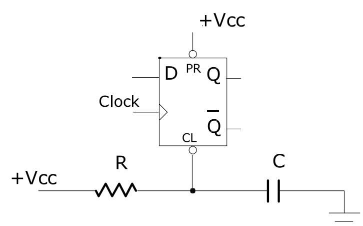


Figura A.55 Uso degli ingressi di $\overline{PR}$ e $\overline{CL}$ per l'azzeramento del FF al momento della *messa sotto tensione*. Nello schema i due ingressi sono attivi bassi. Mentre $\overline{PR}$ passa immediatamente a 1, la rete RC mantiene basso (asserito) l'ingresso $\overline{CL}$ per un tempo dipendente dalla costante RC. Di conseguenza il flip-flop si porta nello stato 0. Quando anche $\overline{CL}$ passa a 1 gli ingressi asincroni $\overline{PR}$ e $\overline{CL}$ diventano ininfluenti e il flip-flop può essere comandato attraverso il clock.

A.8.3 Flip-Flop Master-Slave

Si faccia riferimento alla rete di Figura A.56, nella quale è messa in evidenza la presenza di un FFSR. Vogliamo mostrare che un flip-flop, realizzato come quello in Figura A.51, può dare un serio inconveniente.

Si indichino con Δ_{FF} e con Δ_C i tempi richiesti rispettivamente dal flip-flop e dalla rete combinatoria per commutare. Si supponga che l'ingresso I sia stabile da prima del passaggio a 1 del clock e che resti stabile per tutto Δ_1. Trascorso Δ_{FF} dal fronte di salita, se S e R hanno valori convenienti rispetto allo stato del FF, si attua il cambiamento di stato. A quel punto è possibile che, pur restando I non modificato, il nuovo y determini in uscita da RC – dopo il tempo Δ_C – una coppia (S, R) diversa dalla precedente, la quale, in linea di principio, può comportare un nuovo y. Se ciò si verifica, FFSR di Figura A.56 commuta ulteriormente.

In altri termini si viene a generare una situazione molto diversa da quella ipotizzata al Paragrafo A.8, e cioè: l'ingresso (S, R) non rimane costante durante il Δ_1 del clock.

È facile convincersi che se la rete fosse più complicata, con più livelli combinatori e più livelli di memoria, si potrebbe anche generare una situazione per cui durante Δ_1 un FF cambia più volte stato, con la conseguenza che lo stato finale dipenderebbe in modo

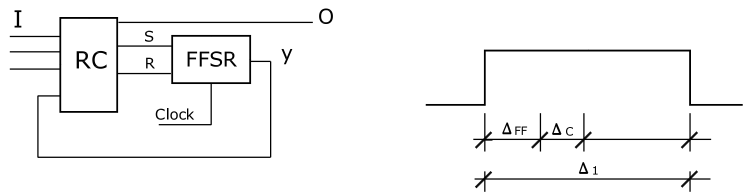


Figura A.56 Rete sequenziale sincrona. L'uscita del flip-flop viene riportata in ingresso alla parte combinatoria
.

impredicibile della durata di Δ_1 e dai tempi di commutazione e potrebbe anche essere diverso da quello implicato dai valori di S e R all'atto del passaggio a uno del clock.

Quello che si vuole da una rete sincrona è che lo stato futuro sia esattamente determinato dalla configurazione d'ingresso e dallo stato della rete all'atto dell'impulso di clock. A tal fine, occorre rivedere il modo in cui i flip-flop commutano. Restando fedeli alla commutazione sui livelli, il problema viene risolto con la configurazione Master-Slave di Figura A.57. In essa sono presenti due latch in cascata: quello a sinistra viene detto *Master*, quello a destra *Slave*. Quando il clock è nello stato 1 il master può cambiare stato, lo slave ha ingressi a zero e quindi non può cambiare stato. Quando il clock passa allo stato 0, lo slave si porta nello stato raggiunto dal master, mentre questo non può più commutare.

Con riferimento alla Figura A.56, deve essere Δ_1 maggiore o uguale al tempo di commutazione del master più lento nella rete, affinché quest'ultimo abbia l'uscita stabile dopo Δ_1 e quindi lo slave abbia ingressi stabili quando si verifica la transizione che porta

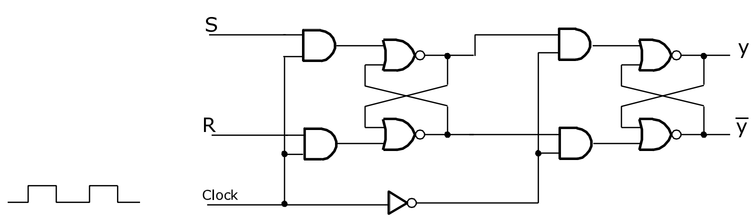


Figura A.57 Configurazione Master-Slave.

il clock a zero.

I precedenti concetti vengono riassunti in Figura A.58. La Figura A.58a rappresenta una generica rete. Per essa si assume che allo scadere di Δ_2 (al momento del passaggio a 1 del clock), l'ingresso I abbia raggiunto una configurazione stabile e che esso non cambi durante Δ_1. Durante Δ_1 i master cambiano stato, ma nessuna delle variabili di stato cambia e, poiché I non cambia, restano immutati gli ingressi ai FF. Dunque, durante Δ_1 nessun segnale cambia entro la rete di Figura A.58a. Questo fatto è schematizzato con la linea continua in Figura A.58b.

Durante Δ_2 i FF presentano le uscite y; è quindi possibile che si modifichino gli ingressi di RC. Di conseguenza nella rete si instaura un transitorio che termina quando

le porte nella rete combinatoria hanno concluso le loro commutazioni. Questo fatto è schematizzato con l'area ombreggiata in Figura A.58b. Naturalmente in questa fase il fatto che cambino gli ingressi ai FF non ha alcuna conseguenza sulle relative uscite. Si richiede solo che gli ingressi ai FF siano stabili prima del termine di Δ_2, in modo che al successivo passagio a 1 del clock, gli elementi di memoria campionino ingressi ben definiti.

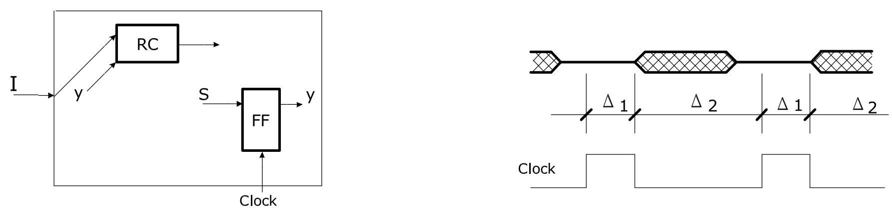


Figura A.58 Generica rete sequenziale sincrona e cambiamenti dei segnali.

È chiaro ora che anche l'ingresso primario I deve cambiare entro Δ_2, e che Δ_2 deve essere sufficientemente lungo da permettere la commutazione della parte combinatoria. In conclusione, si hanno questi vincoli per Δ_1 e Δ_2:

- Δ_1 viene scelto in modo da essere maggiore al tempo di commutazione del più lento dei master;
- Δ_2 viene scelto in modo da garantire la commutazione della parte combinatoria. L'ingresso primario I deve cambiare entro Δ_2, in modo da garantire il rispetto di questa condizione. Ciò sicuramente accade se I è pure sincronizzato con il clock, ovvero se I è l' uscita di FF Master-Slave pilotati dal medesimo clock.

A.8.4 Commutazione sui fronti.

Lo schema di funzionamento del flip-flop Master-Slave ci ha permesso di ragionare secondo i canoni della logica booleana, ovvero tenendo in conto solo i livelli dei segnali. È ben noto che larga parte dei FF commuta sui fronti di salita o di discesa del clock. Gli stessi FF Master-Slave sono costruiti in modo da effettuare la commutazione del master sul fronte di salita e quello dello slave sul fronte di discesa.

È evidente che la commutazione sui fronti è preferibile alla commutazione sui livelli. Infatti, per il loro corretto funzionamento è sufficiente che gli ingressi siano stabili sui fronti. Anche se di durata non nulla, un fronte è comunque un tempo molto breve. Nella pratica bisogna tuttavia tener conto che c'è un intervallo di tempo a cavallo del fronte di commutazione durante il quale gli ingressi devono rimanere stabili. Questo intervallo è diviso in due parti: il tempo di *set up*, prima del fronte di commutazione e il tempo di *hold*, dopo il fronte di commutazione. Il funzionamento sul fronte ha due vincoli (Figura A.59):

a) il tempo di discesa/salita del fronte deve essere non superiore ad un limite massimo;
b) cavallo del fronte di commutazione l'ingresso deve rimanere stabile. Questo intervallo è diviso in due parti: il tempo di *set up* (τ_S), prima del fronte di commutazione e il tempo di *hold* (τ_H), dopo il fronte di commutazione.

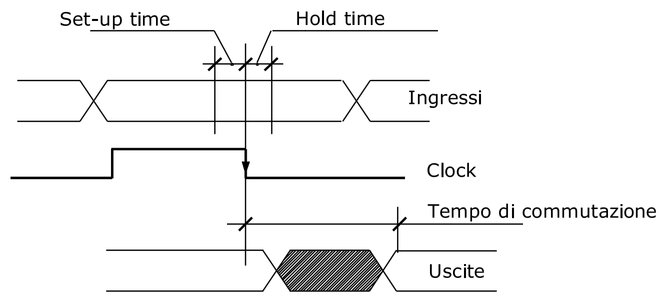


Figura A.59 Schematizzazione della commutazione di un flip-flop a commutazione sul fronte (*edge-triggered*) di discesa. È necessario che gli ingressi al flip-flop siano stabili per almeno il tempo di *set-up* (τ_S) prima del fronte che determina la commutazione e che vengano mantenuti stabili per il tempo di *hold* (τ_H) dopo il fronte. Il tempo di commutazione (τ_C) è quello che intercorre tra il fronte del clock e l'istante in cui le uscite sono stabili. Si noti che $\tau_C > \tau_H$.

Vale la pena di osservare che con l'ipotesi di commutazione sul fronte si ha una semplificazione concettuale notevole, anche rispetto al caso di flip-flop Master-Slave. Infatti, il periodo minimo del clock deve essere tale da garantire la propagazione dei segnali, oltre ai due tempuscoli attorno al clock. Ovviamente i segnali primari in ingresso alla rete possono cambiare in qualunque momento tra due clock, purché non inducano una situazione di instabilità in prossimità del fronte. Quando i flip-flop commutano sul fronte, il clock viene semplicemente rappresentato come una freccia verso l'alto o il basso, a seconda del fronte che determina la commutazione.

Avvertenza.
Data la semplicità concettuale della commutazione comandata dal fronte, nel seguito assumeremo sempre – a meno di avviso contrario – che questo sia il caso, anche quando il clock viene rappresentato nel modo usuale.

Differenze tra modello di Mealy e modello di Moore

Col modello di Moore le uscite sono funzione del solo stato e siccome lo stato cambia in corrispondenza degli impulsi di clock, anche le uscite risultano sincronizzate rispetto al clock. Col modello di Mealy le uscite, essendo funzione anche degli ingressi, possono variare in modo asincrono rispetto al clock. Spieghiamo la differenza con un esempio.

Supponiamo di dover progettare la rete R di Figura A.60, a un solo ingresso e una sola uscita, che deve esibire questo comportamento: quando l'ingresso è zero, la rete deve generare in uscita impulsi di durata pari a un periodo di clock; l'uscita deve essere sincronizzata rispetto al clock.

Per avere le uscite sincronizzate occorre fare riferimento al modello di Moore. In Figura A.61 vengono riportati il diagramma di stato, la tabella di flusso e la tabella delle transizioni (con la codifica A = 0, B = 1) corrispondenti alla specifica. Si trova immediatamente

$$z = y \qquad\qquad y' = \overline{x}\,\overline{y}$$

per cui è facile tracciare lo schema logico.

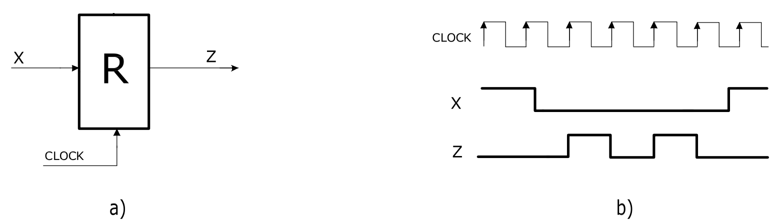


Figura A.60 Specifica di una rete sequenziale con uscita sincrona: a) schema di progetto; b) relazioni temporali da rispettare (l'uscita z deve essere sincronizzata con il clock).

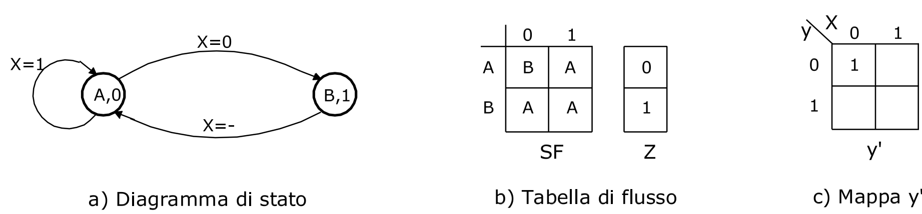


Figura A.61 Sintesi della rete R secondo il modello di Moore.

Supponiamo ora di rilasciare il vincolo secondo cui l'uscita deve essere funzione del solo stato e vediamo quale sarebbe stato il risultato della sintesi ricorrendo al modello più generale di Mealy. In Figura A.62 vengono riportati i corrispondenti diagramma di stato, tabella di flusso e delle transizioni. Da quest'ultima si ricava

$$z = \overline{x}\,\overline{y} \qquad\qquad y' = \overline{x}\,\overline{y}$$

Ciò che cambia è che z è stato posto a 1 solo in corrispondenza dello stato A e dell' ingresso 0. Conseguentemente l'uscita z può avere l'anomalia nel passaggio di x da alto a basso, come illustrato in Figura A.63. Dopo il primo clock la rete si comporta correttamente.

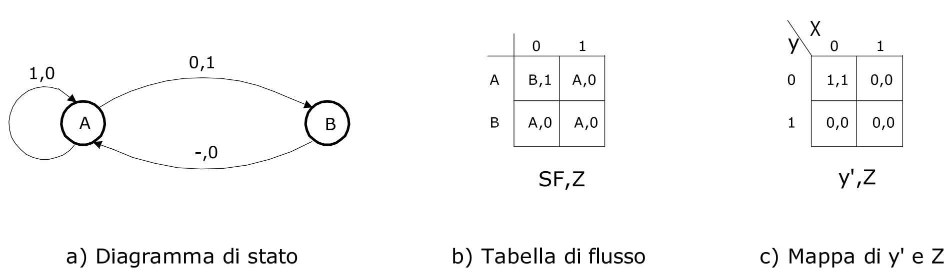


Figura A.62 Sintesi della rete R secondo il modello di Mealy.



Figura A.63 Rete secondo il modello di Mealy e possibile sequenza temporale. Si noti che appena x passa a 0 la rete asserisce l'uscita z. Ma l'uscita, sul primo clock che campiona x a 0, durerà normalmente meno di un periodo; in media metà periodo.

A.9 Reti sequenziali sincrone

Nel modello di Figura A.47 gli elementi τ_i introducono ritardi potenzialmente differenti, la cui genesi era dovuta ai differenti tempi di commutazione sui vari percorsi seguiti dai segnali. Se ora si immagina di sostituire gli elementi τ_i con altrettanti FFD e si suppone che tutti questi vengano azionati dal medesimo clock, il modello di Figura A.47 si trasforma nella rete sequenziale sincrona come quella di Figura A.64.

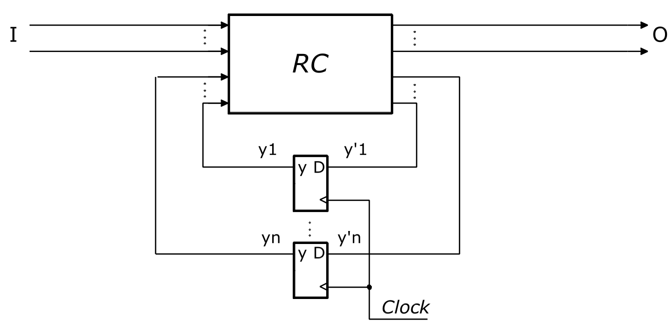


Figura A.64 Modello di rete sequenziale sincrona.

La rete di Figura A.64 cambia stato solo in corrispondenza degli impulsi di clock. Non si richiede più che la rete combinatoria sia ideale. Ipotizzando che i flip-flop lavorino sul fronte di discesa, la rete si comporta in questo modo:

- A partire da uno stato stabile S, al variare di I in uscita a RC si produce una nuova configurazione; le uscite $y'_1, \cdots y'_n$ (ovvero lo stato futuro S') in ingresso ai flip-flop dovranno essere stabili prima del fronte del clock in accordo alla schematizzazione di Figura A.59.
- Sul fronte del clock i FF campionano i propri ingressi e portano la rete nel (possibile) nuovo stato; dopo il tempo di hold, il nuovo stato (necessariamente stabile) appare in ingresso a RC e quindi, potenzialmente, l'uscita O può ancora cambiare. Passato il transitorio, l'uscita è quella che corrisponde alla relazione $O = f(I, S)$, dove $S = S'$ è il nuovo stato.

Nello schema di Figura A.64, si sono usati FFD. Ovviamente può essere impiegata qualunque combinazione di FF.

Differenze tra reti sincrone e asincrone

Ai Paragrafi A.7 e A.7.2, illustrando le reti sequenziali asincrone si era parlato di *stato stabile*. Con le reti sincrone e flip-flop Master-Slave il concetto di stato stabile viene a perdere di significato. Infatti, per quanto detto sopra, la rete campiona i propri ingressi e si porta in un nuovo stato prima dell'arrivo del prossimo clock. Dunque la rete, se ben progettata, passa sempre attraverso stati stabili. Naturalmente ciò non esclude che al clock successivo ci sia un ulteriore cambiamento di stato.

Analizzando le reti asincrone si era inoltre stabilito che gli ingressi dovessero variare uno alla volta, al fine di evitare comportamenti impredicibili. Questo vincolo non ha più ragione di essere per le reti sincrone che rispettano la modalità di funzionamento sopra illustrata, in quanto il clock arriva quando i segnali di ingresso sono tutti stabili, evitando i fenomeni di *corsa*.

In conclusione le reti sincrone sono molto più semplici e affidabili delle asincrone. Per questo nelle parti del testo precedenti questa appendice si è sostanzialmente fatto riferimento alle reti sincrone. Come illustrato nel testo, specialmente ai capitoli sulla CPU e sulla pipeline, i sistemi di elaborazione sono delle gigantesche reti sincrone, a cui il clock dà il ritmo di esecuzione.

A.9.1 Aspetti di organizzazione delle reti sincrone

Nella progettazione di un sistema digitale complesso è spesso necessario dividere la logica in un certo numero di stadi operanti in cascata, facendo in modo che una data funzionalità venga eseguita in più cicli di clock. A tale scopo si rende necessario "disaccoppiare" il funzionamento delle sottoreti, nel senso che l'ingresso di una sottorete a valle deve essere determinato solo dalla sottorete a monte e non dall'ingresso primario. Ciò è garantito quando le uscite delle sottoreti sono funzione del solo stato, come previsto dal modello di Moore.

Un altro modo per ottenere le uscite sincrone, consiste nell'interposizione di un numero adeguato di flip-flop, come schematizzato in Figura A.65.

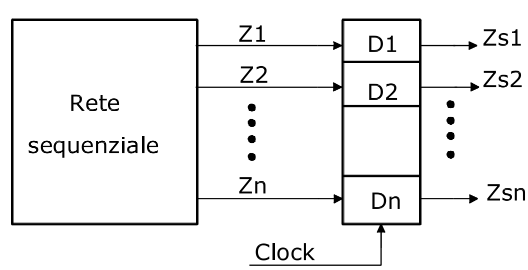


Figura A.65 Sincronizzazione delle uscite tramite interposizione di un registro di transito.

Tali flip-flop possono essere riguardati come un *registro di transito*. La rete sequenziale che precede il registro può essere anche asincrona o, meglio ancora, una pura rete combinatoria, che, tra un clock e il successivo, elabora i suoi ingressi (le uscite del registro a monte) e presenta al registro a valle, che li memorizza sul clock, i risultati dell'elaborazione. Si ottiene così lo schema di Figura A.66.

Di norma, il periodo minimo del clock della rete a stadi, come quella di Figura A.66, risulta più breve di quello che si avrebbe con un'unica rete, in quanto l'intervallo di tempo

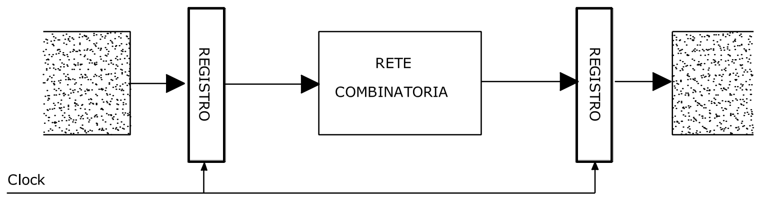


Figura A.66 Modello di rete di elaborazione. La rete combinatoria deve elaborare i segnali in ingresso e presentarli al registro in uscita in modo che vengano memorizzati sul prossimo fronte (di commutazione) del clock. Si noti che questa schematizzazione corrisponde esattamente all'organizzazione delle pipeline (Capitolo 9 del testo).

richiesto dalla logica combinatoria dei singoli stadi risulta minore di quanto richiederebbe una rete non a stadi. Ma il tempo complessivo per eseguire una data funzione è superiore.

Infatti, sia T il periodo di clock richiesto da una rete non a stadi e T' quello richiesto da uno stadio (che supporremo, per semplicità, uguale per tutti gli stadi). Ci si aspetta che sia $T' < T$, essendo la parte combinatoria degli stadi più piccola di quella della rete non a stadi. Ma la rete a stadi opera in un tempo nT' (dove n è il numero di stadi), dunque, di norma, si avrà $T < nT'$.

A.10 Registri

Per registro si intende un insieme di n identici elementi di memoria (flip-flop), sincronizzati tramite un unico clock. I registri sono componenti essenziali dei sistemi di elaborazione in quanto costituiscono i supporti che materializzano l'informazione nella macchina. Inoltre, come illustrato in precedenza, permettono di partizionare la logica complessiva e di scomporla in blocchi semi-indipendenti.

I registri possono differenziarsi per il tipo di flip-flop impiegati, per il modo in cui essi vengono comandati, per le funzioni aggiuntive a quelle di contenitore dell'informazione (per esempio, lo scorrimento, il conteggio ecc.).

Per fissare le idee facciamo riferimento allo schema di Figura A.67. Il registro in questione è formato di FFSR o FFJK. Il funzionamento del registro è il seguente.

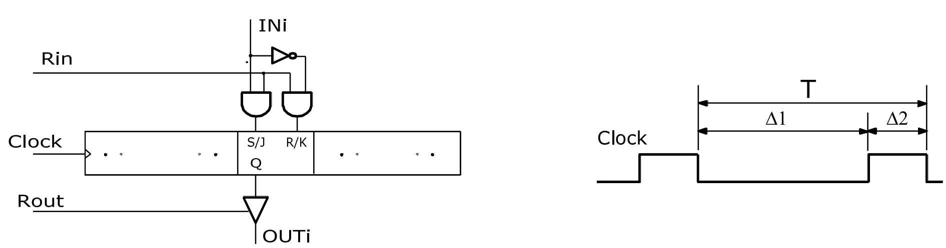


Figura A.67 Struttura di un registro. Viene mostrato il dettaglio per il generico bit. Si assume che il registro commuti sul fronte di discesa del clock. Il segnale R_{out} abilita l'uscita in terzo stato.

- Se R_{in} è disasserito, l'ingresso ai singoli flip-flop è 00 e, dunque, il registro si mantiene nello stato precedente; se R_{in} è asserito, sul prossimo clock lo stato di ogni flip-flop (il contenuto del registro) diventa quello corrispondente all'ingresso INi.
- Il segnale R_{out} ha funzione di *Output Enable* dell'uscita a tre stati.

I segnali R_{in} e R_{out} svolgono la funzione di segnali di controllo o di comando. Con lo schema di Figura A.67, il clock è ininfluente quando $R_{in} = 0$. Se si fosse usato FFD, il clock avrebbe dovuto essere abolito quando $R_{in} = 0$[20]. In Figura A.68 si riporta la schematizzazione impiegata nel seguito.

.

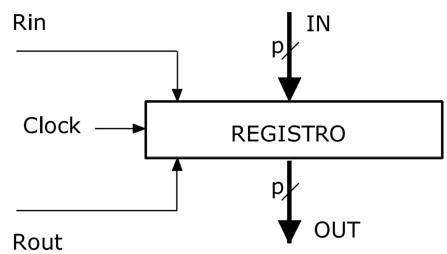


Figura A.68 Modello di registro. A seconda della convenienza, spesso si omette di indicare i segnali R_{in} e R_{out}; in tal caso si deve assumere che essi sono sempre asseriti. Si omette pure di indicare il clock, che pure va inteso come sempre presente, a meno di avviso contrario. Il numero p di linee binarie in ingresso o in uscita è pari alla dimensione (parallelismo) del registro.

Caricamento asincrono.
A volte si presenta la necessità di caricare un registro in modo asincrono indipendentemente dal meccanismo legato al clock, illustrato in precedenza. Ad esempio, può esserci la necessità di azzerare eccezionalmente il contenuto del registro tra un clock e l'altro. Al Paragrafo A.8.2 è stato illustrato l'impiego degli ingressi asincroni Preset (PR) e Clear (CL). In Figura A.69 viene riportato lo schema del caricamento asincrono di un flip-flop.

Registri a scorrimento.
Un registro a scorrimento prevede i segnali di controllo per comandare lo scorrimento verso destra o verso sinistra di uno o più bit. Il bit estremo dalla parte verso cui avviene lo scorrimento è perso, mentre il bit estremo dalla parte da cui inizia lo scorrimento deve essere alimentato nel modo seguente:

a) poiché scorrimento verso sinistra di una posizione equivale a moltiplicare per 2 il numero contenuto nel registro, si richiede che nel bit meno significativo venga inserito uno 0[21];

[20] Nella costruzione di reti complesse (sincrone) è preferibile che tutti i componenti ricevano sempre il clock, evitando l'aggiunta di porte per silenziarlo. Per questo è preferita la soluzione di Figura A.67. Il risultato equivalente si otterrebbe sfruttando R_{in} come segnale di clock ai FFD.

[21] È possibile che scorrendo verso sinistra un numero positivo possa diventare negativo e viceversa. Per evitare cambiamenti di segno è necessario tenere bloccato il bit più significativo, lasciando scorrere gli altri.

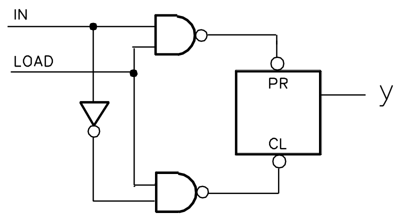


Figura A.69 Caricamento asincrono del dato IN nel flip-flop tramite il comando LOAD.

b) poiché lo scorrimento verso destra di una posizione equivale a dividere per 2, è necessario che nel bit più significativo venga introdotta la copia del segno (cioè copia del medesimo bit).

Registri ad anello.
Quando un registro a scorrimento è organizzato in modo che, a seconda della direzione di scorrimento (verso sinistra o verso destra), il bit più significativo si porti nel bit meno significativo o viceversa, si ha un registro di scorrimento ad anello (ovvero si ha uno scorrimento circolare o rotazione).

Se ora supponiamo che un solo bit di un registro ad anello di n bit sia a 1 e osserviamo le uscite dei singoli FF, queste sono 1 per un periodo di clock su n. In altre parole, si può usare un registro a scorrimento per generare n identici segnali temporizzati, con fase diversa. Se f è la frequenza del clock, si hanno n segnali di periodo $T = n/f$, sfasati tra loro di T/n, come illustrato in Figura A.70.

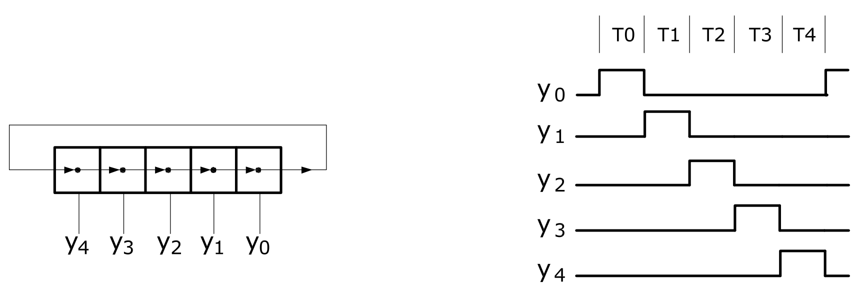
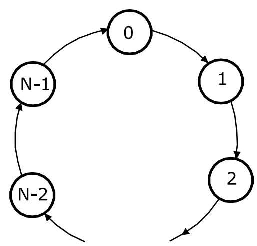


Figura A.70 Registro ad anello (di 5 bit) e forme d'onda corrispondenti allo stato dei singoli bit. Il registro è precaricato in modo da contenere un solo 1.

Un registro come quello ora discusso svolge a tutti gli effetti la funzione di contatore modulo n. L'uso di un registro ad anello all'interno della logica di CPU è stato illustrato nel testo al capitolo sulla CPU. Paragrafo 2.8.1 del testo.

Contatori.
Un contatore è una rete sequenziale con un solo ingresso costituito da un segnale, normalmente periodico (il clock), del quale si vogliono contare gli impulsi. Nota la frequenza del clock, il conteggio del numero di impulsi equivale alla misura del tempo.

Se si fa riferimento a quanto detto in precedenza circa le reti sincrone, per le quali il clock non viene considerato come un ingresso, ma come parte integrante del loro funzionamento, i contatori costituiscono una classe di reti degeneri in quanto prive di ingresso. La loro attività consiste nel cambiare stato (avanzare il conteggio) ad ogni impulso di clock. Indicando con N il numero di stati interni, il diagramma di stato di un contatore è come quello rappresentato in Figura A.71, dove gli archi non sono contrassegnati in quanto le transizioni avvengono comunque su ogni impulso di clock. Un registro formato da n flip-flop può arrivare a contare modulo $N = 2^n$.



SP	SF
0	1
1	2
2	3
...	...
N-2	N-1
N-1	0

Figura A.71 Diagramma di stato e tabella di flusso di un contatore modulo N. SP indica lo stato presente ed SF lo stato futuro. Il numero N, corrispondente al numero di stati interni, viene detto modulo del contatore.

Contatori sincroni.

Si parla di contatori sincroni se tutti i flip-flop ricevono lo stesso clock e quindi commutano sullo stesso istante.

In Figura A.72 viene riportato lo schema di un contatore sincrono modulo 4, realizzato con FFD e FFJK.

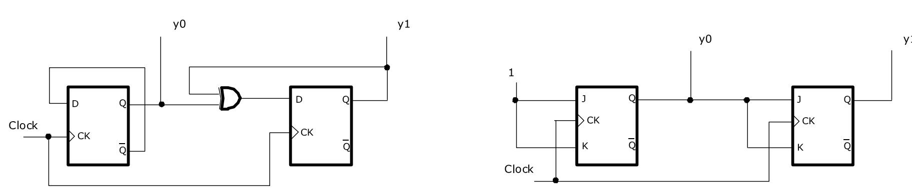


Figura A.72 Schemi realizzativi di un contatore modulo 4 con FFD e FFJK.

Lo schema di Figura A.72 è facilmente generalizzabile al caso del contatore modulo 2^n. Basta osservare che il flip-flop i_{mo}, con $0 < i \leq n-1$, commuta quando tutti i FF da 0 a $i-1$ sono ad 1. Da ciò deriva $D_i = (y_0 y_1 \cdots y_{i-1}) \oplus y_i$, per contatori con FFD; e $J_i = K_i = y_0 y_1 \cdots y_{i-1}$ per contatori con FFJK. Si osservi che la variabile y_i è un segnale periodico con frequenza pari alla frequenza del clock divisa per 2^{i+1} (si veda il diagramma temporale di Figura A.73).

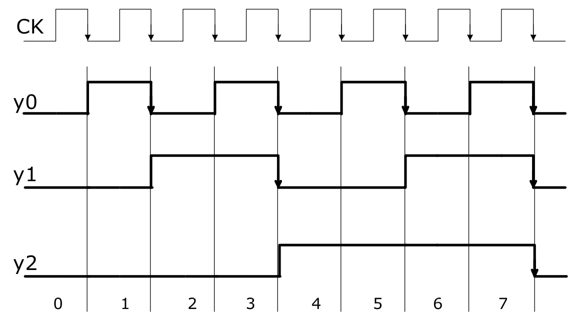


Figura A.73 Temporizzazione dei segnali corrispondenti agli stati dei FF di un contatore modulo 8 (registro di 3 bit). Come si vede lo stato di ciascun flip-flop è un segnale periodico con una frequenza dimezzata rispetto a quella del precedente.

Contatori asincroni.

Si parla di contatori asincroni se non tutti i flip-flop ricevono l'impulso da conteggiare.

L'impulso di conteggio non viene inviato a tutti i FF, ma al solo FF corrispondente al bit meno significativo. I FF successivi ricevono impulsi ottenuti con opportune combinazioni delle uscite precedenti. A titolo di esempio, in Figura A.74 viene riportato lo schema di un contatore asincrono modulo 6. Si noti che senza la parte di rete a destra il contatore avrebbe contato modulo $2^3 = 8$. Questa parte di rete serve a riconoscere la configurazione 6 e a riportare i tre FF a 0 asserendo gli ingressi CL dei FF stessi. Il latch interposto tra la decodifica del 6 e il contatore evita che differenti velocità di commutazione dei FF inducano un comportamento errato[22]

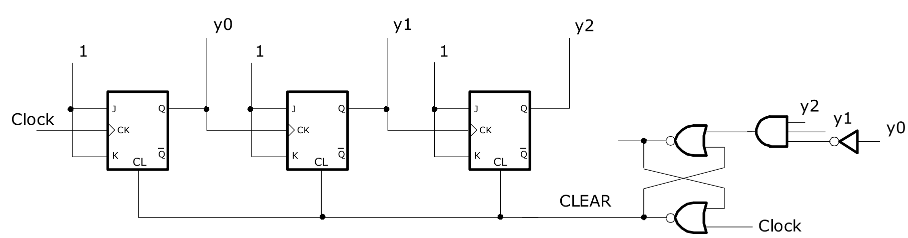


Figura A.74 Contatore asincrono modulo 6. I flip-flop commutano sul fronte di discesa. Il latch mantiene asserito il Clear durante la fase bassa del clock.

[22] Senza il latch il segnale Clear resterebbe asserito solo finché permane il 6 sulla porta AND. Se un FF è piu lento degli altri e non si azzera in tale lasso di tempo, il contatore non funziona correttamente.

A.11 Trasferimento dell'informazione

Larga parte delle attività all'interno di un calcolatore consiste nel trasferimento dei loro contenuti tra i registri.

Per trasferire l'informazione da un registro all'altro occorre che l'uscita del registro sorgente venga portata all'ingresso del registro di destinazione. Se si hanno m possibili registri sorgente e n possibili registri di destinazione, in linea di principio si dovrebbero avere $m \times n$ percorsi e una logica di selezione che permetta di scegliere i percorsi che interessano in un dato istante. Una simile rete è componibile attraverso porte AND/OR

Una rete di $m \times n$ percorsi rende possibile trasferimenti multipli su ciascun clock, compreso il trasferimento da uno a più registri e il trasferimento dell'OR del contenuto di più sorgenti in una o più destinazioni (si veda l'Esercizio A.23). Tuttavia una simile rete ha lo svantaggio della complessità e dell'occupazione di spazio (sul *chip* o sulla piastra) per cui si preferisce ricorrere alla struttura a bus.

A.11.1 Struttura a bus

Sebbene sia possibile costruire un bus con sole porte AND/OR (si veda l'Esercizio A.24), la soluzione migliore è realizzare i bus con logica in terzo stato, eventualmente con logica a collettore aperto. Lo schema di un tale bus è riportato in Figura A.75. Il trasferimento dal registro RSi al registro RSj richiede che siano asserite le due linee di comando BUS←RSi e RDj←BUS. Si è omesso di indicare il clock comune a tutti i registri.

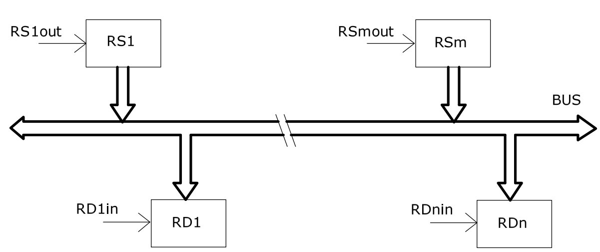


Figura A.75 Schema di collegamento tramite bus. Si suppone le uscite dei dispositivi collegate sul bus siano a tre stati. In un dato momento deve essere asserito un solo segnale di abilitazione delle uscite, in modo che il bus venga portato allo stato logico del corrispondente registro. I segnali di abilitazione degli ingressi determinano quali registri di destinazione vengono caricati sul prossimo (fronte del) clock.

Se le uscite sul bus sono in logica a tre stati solo un trasmittente alla volta può avere l'uscita abilitata. Nel caso che nessun dispositivo sorgente abbia l'uscita abilitata il bus viene a trovarsi in uno stato indeterminato e l'eventuale trasferimento in uno o più registri di destinazione è indeterminato esso stesso.

Se le uscite sono a collettore aperto, è possibile abilitare più di una uscita alla volta; in tal caso si ha il *wired* AND (logica positiva) delle uscite stesse. Con le uscite a collettore aperto lo stato del bus non è mai indeterminato. Nel caso estremo in cui le uscite di tutti i dispositivi sul bus siano interdette, lo stato del bus è a livello logico alto, per via delle (necessarie) resistenze di *pull-up* (A.4, Figura A.22).

La struttura a bus è lo standard nel mondo dei calcolatori, ma ha un inconveniente: consente un solo trasferimento alla volta. È questo un fattore che tende a trasformare il bus nel "collo di bottiglia" del sistema. Inoltre, in presenza di più unità indipendenti che possono comandare l'uso del bus, si richiede una logica di arbitraggio che consenta di risolvere le contese in caso di richieste di accesso concomitanti. Nei sistemi ad alte prestazioni, con molti sottosistemi interagenti tra di loro (processori, memorie, ..), è necessario ricorre a più complesse reti di interconnessione.

A.11.2 Tempificazione

Vogliamo ora analizzare più nel dettaglio la tempificazione di un trasferimento dell'informazione tra due registri attraverso un bus e determinare il periodo minimo per il clock.

In Figura A.76 si mette in evidenza la parte di controllo, rappresentata dal blocco CNTRL, responsabile della generazione dei due comandi RS_{out} e RD_{in}; tutto il sistema è comandato dallo stesso clock. La durata minima del periodo di clock deve essere tale da garantire che al generico clock l'ingresso al registro RD sia stabile per oltre il suo tempo di setup. Dunque, si deve tener conto di questi tempi (si faccia riferimento alla Figura A.77):

- τ_g : tempo richiesto dalla logica CNTRL per generare segnali RS_{out} e RD_{in} dall'istante in cui si ha il fronte del clock i;
- τ_1 : tempo impiegato dal segnale RS_{out} a propagarsi da CNTRL a RS;
- τ_{out} : tempo richiesto per il passaggio in conduzione dell'uscita di RS;
- τ_B : tempo di trasmissione del dato da RS fino a RD attraverso il bus;
- τ_S : tempo di setup dei flip-flop del registro di destinazione;
- τ_H : tempo di hold dei flip-flop del registro di destinazione.

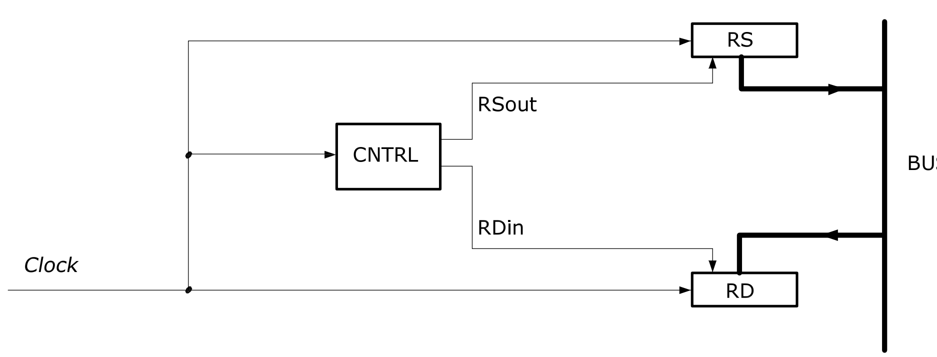


Figura A.76 Esempio di trasferimento da un registro ad un altro attraverso il bus. Il blocco CNTRL è responsabile della generazione dei segnali RS_{out} e RD_{in}.

Indicando con T il periodo di clock, deve essere soddisfatta la seguente relazione:

$$T \geq \tau_g + \tau_1 + \tau_{out} + \tau_B + \tau_S + \tau_H \tag{A.15}$$

Tuttavia, il periodo T può essere ridotto sovrapponendo l'attività del registro di destinazione con quella del trasferimento dal registro sorgente a quello di destinazione stesso, ovvero se

$$T = \tau_g + \tau_1 + \tau_{out} + \tau_B \geq \tau_S + \tau_H \tag{A.16}$$

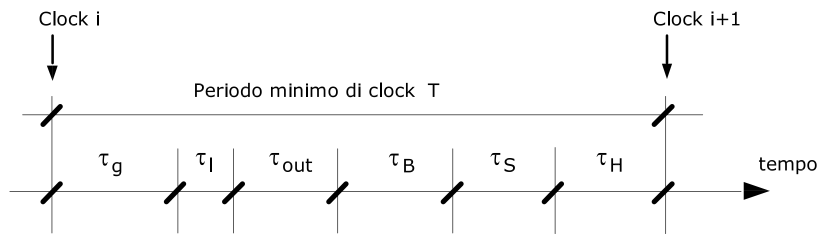


Figura A.77 Quantificazione del periodo minimo del clock imposto dal segnale RS$_{out}$ nel trasferimento di Figura A.76, nel caso e non vi sia nessuna sovrapposizione tra la fase di trasferimento dal registro sorgente a quello di destinazione e la fase di commutazione del registro di destinazione.

Approfondimento: Progetto di reti sequenziali

La sintesi di una rete sequenziale, ovvero il processo che porta da una specifica della rete in forma testuale al corrispondente schema logico, si compone di più passi:

a) tracciatura del diagramma di stato e della corrispondente tabella di flusso;
b) minimizzazione del numero degli stati;
c) codifica degli stati attraverso variabili di stato booleane;
d) determinazione delle funzioni di eccitazione dei flip-flop;
e) determinazione della parte combinatoria rimanente.

Il primo passo è quello che il progettista compie in base alla specifica, interpretando quanto da essa imposto. Si tratta di un processo creativo, che deve comunque portare a definire un diagramma di stato che tenga conto di tutte le possibili condizioni in cui la rete può venire a trovarsi. La stesura di un n semplice di diagramma di stato è stata mostrata all'A.8.4. Nel tracciare il diagramma di stato il progettista deve puntare a tener conto di tutte le possibili condizioni in cui la rete si può, a suo giudizio, venire a trovarsi, anche a rischio di introdurre stati ridondanti. Questi verranno eliminati col passo successivo.

Il secondo passo consiste nell'accorpare stati che sono *equivalenti* (o *compatibili*), ovvero che rappresentano la medesima condizione di stato della rete. Il processo si attua a partire dalla tabella di flusso di partenza, tramite un metodo algoritmico che porta alla tabella di flusso minima (alla quale corrisponde, evidentemente, una diagramma degli stati minimo). L'illustrazione del processo di minimizzazione è aldilà dello scopo di questo libro e non verrà presa in considerazione. Del resto, in tutti i casi di esempio presentati nel testo, il diagramma di stato primitivo è sempre anche quello minimo.

Il passo c) consiste nel codificare gli stati simbolici della tabella di flusso minima con un numero conveniente di variabili binarie di stato. Se la tabella ha N righe, allora occorrono almeno un numero di variabili di stato (e quindi di flip-flop) pari all'intero immediatamente superiore o uguale a $\log_2 N$. La tabella che si ottiene sostituendo agli stati simbolici della tabella di flusso le loro codifiche viene detta *tabella delle transizioni*. Essa fornisce le funzioni di stato futuro e le funzioni di uscita della rete.

Il quarto passo richiede che si determinino gli ingressi ai flip-flop che realizzano le variabili di stato. Ciò si fa a partire dalle funzioni di stato futuro di ciascun flip-flop, tenuto conto delle specifiche caratteristiche di funzionamento dei flip-flop scelti. In tal modo risulta determinata la parte combinatoria che porta gli ingressi primari e le variabili di stato agli ingressi dei flip-flop.

L'ultimo passo consiste nel determinare la parte combinatoria relativa alle funzioni di uscita.

Funzioni di eccitazione dei flip-flop

Nella determinazione degli ingressi agli elementi di memoria della rete occorre tener conto del modo di funzionamento dei flip-flop impiegati. Mostreremo come, riferendoci alla funzione di stato di Figura A.78.

Flip-Flop D

Nel caso dei questo flip-flop lo stato futuro corrisponde all'ingresso presente, ovvero $y' = D$; in altre parole, l'ingresso D è la realizzazione della funzione y'. Ne consegue lo schema di Figura A.79.

Flip-flop SR e JK

Nel caso di questi flip-flop occorre dare una coppia di ingressi tale da realizzare la tran-

y \ $x_1 x_2$	00	01	11	10
0	1	0	1	1
1	1	1	0	0

y'

Figura A.78 Funzione di stato futuro presa come esempio.

y \ $x_1 x_2$	00	01	11	10
0	1	0	1	1
1	1	1	0	0

$$D = \overline{x_1}\, y + \overline{x_1}\, \overline{x_2} + x_1\, \overline{y}$$

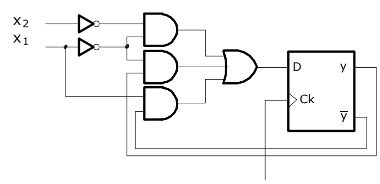


Figura A.79 Uso del flip-flop D per realizzare la variabile di stato futuro y' di Figura A.78.

sizione di stato prevista sulla mappa. In Figura A.80 si riportano le coppie di ingressi richiesti per effettuare le corrispondenti transizioni.

$y \to y'$	S	R
$0 \to 0$	0	–
$0 \to 1$	1	0
$1 \to 0$	0	1
$1 \to 1$	–	0

$y \to y'$	J	K
$0 \to 0$	0	–
$0 \to 1$	1	–
$1 \to 0$	–	1
$1 \to 1$	–	0

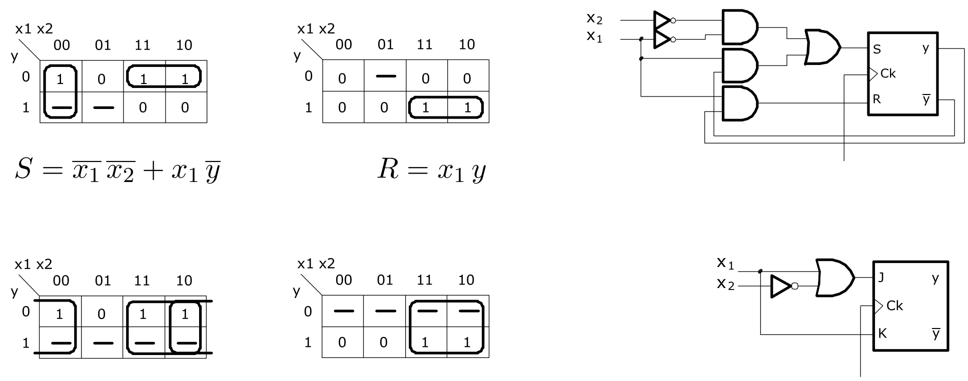
Figura A.80 Ingressi richiesti per le possibili transizioni di stato per i flip-flop SR e JK.

Per ottenere l'espressione degli ingressi occorre modificare la mappa di y', passando alle mappe di S e R, ovvero J e K a seconda del tipo usato. Nel caso del flip-flop SR la mappa di Figura A.78 si trasforma come nella parte superiore di Figura A.81. Nel caso del flip-flop JK la mappa di Figura A.78 si trasforma come nella fascia bassa di Figura A.81. È evidente la convenienza del FFJK.

Esempio: generazione di un segnale di WAIT

Viene ora trattato un problema tipico dei sistemi di elaborazione: la generazione di un segnale tempificato verso la CPU.

Capita spesso che la velocità della CPU è superiore a quella consentita dalla memoria o dai dispositivi di ingresso/uscita, nel senso che i tempi dei cicli di lettura o scrittura



$$S = \overline{x_1}\,\overline{x_2} + x_1\,\overline{y} \qquad R = x_1\,y$$

$$J = x_1 + \overline{x_2} \qquad K = x_1$$

Figura A.81 Uso dei flip-flop SR (fascia superiore) JK (fascia inferiore) per realizzare la variabile di stato futuro y' di Figura A.78. Le tabelle per S e R, e per J e K, sono state ottenute a partire dalla mappa di Tabella A.78, effettuando le sostituzioni come previsto dalla tabella di Figura A.80. Da esse conseguono le reti a destra.

da parte della CPU sono più bassi di quelli richiesti dalla memoria per effettuare tali operazioni. Di solito le CPU, e in particolare quelle disponibili in forma integrata come microprocessori, dispongono di una linea di ingresso, detta WAIT che, se asserita durante l'esecuzione di un ciclo di lettura o scrittura, fa allungare la durata del ciclo in questione per tutto il tempo in cui la linea viene mantenuta asserita (si veda al Capitolo 2 del testo la temporizzazione della fase di fetch).

In Figura A.82 viene dato lo schema di principio. La rete GENW genera un segnale di WAIT, della durata di uno o più periodi di clock, quando viene asserito l'ingresso $\overline{x}$. Nello schema di figura il blocco GX rappresenta un componente del sistema (memoria o periferica di ingresso/uscita) che richiede la generazione del segnale WAIT. Nel caso della memoria, la linea $\overline{x}$ passa allo stato di asserito a seguito della generazione da parte della CPU di un indirizzo relativo a blocco di memoria lento (rispetto alla CPU).

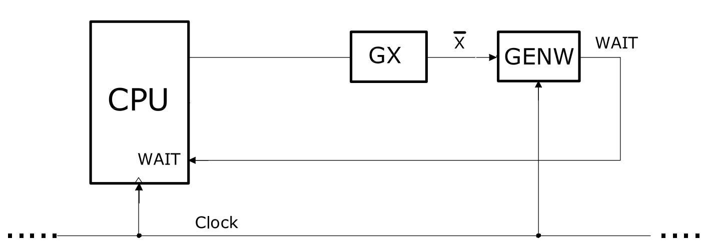


Figura A.82 Schema di principio per la generazione di un segnale di WAIT.

Si deve progettare la rete GENW che soddisfa le seguenti specifiche:

a) quando x è alto, WAIT è basso;
b) quando x viene campionato basso, l'uscita WAIT passa allo stato alto con un ritardo di un periodo di clock;

c) per generare WAIT non è necessario che x venga campionato basso anche sul secondo clock, basta che sia tale una sola volta;
d) WAIT resta a 1 esattamente per un periodo di clock;
e) dopo che WAIT è tornato basso occorre che x ritorni alto prima che sia possibile la generazione di un altro WAIT (al nuovo campionamento di x basso).

La precedente specifica viene sintetizzata dal diagramma temporale di Figura A.83.

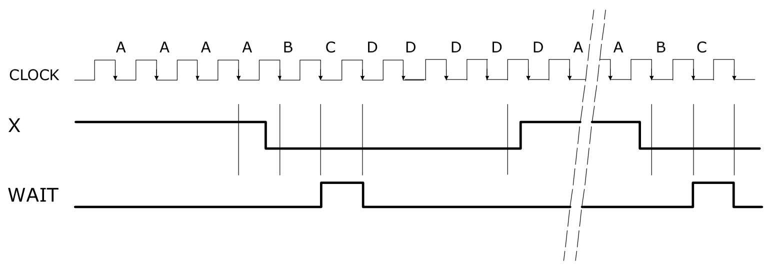
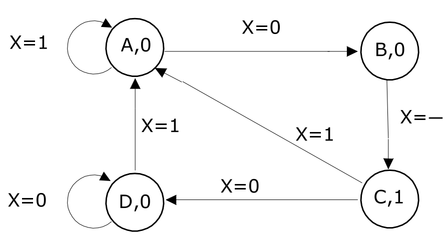


Figura A.83 Diagramma temporale per il WAIT. Il diagramma riporta in corrispondenza del clock lo stato in cui si trova la rete (secondo il diagramma di stato di Figura A.84). Si assume che i FF commutino sui fronti di discesa e che conseguentemente il campionamento di x avvenga sui medesimi fronti.

La specifica impone che l'uscita WAIT duri esattamente un periodo di clock: ciò viene garantito rendendo l'uscita sincrona rispetto al clock, ovvero rendendo l'uscita funzione del solo stato della rete. Occorre quindi riferirsi al modello di Moore. In Figura A.84 è stato disegnato il diagramma di stato deducibile dalle specifiche.



	$x=0$	$x=1$
A	B	A
B	C	C
C	D	A
D	D	A

SF

–
0
0
1
0

WAIT

Figura A.84 Diagramma degli stati della macchina di Moore che sintetizza le specifiche del generatore di WAIT e relativa tabella di flusso. Lo stato A corrispondente è quello in cui si trova la rete a riposo. Al passaggio di x a zero la rete si porta nello stato B, dove l'uscita è mantenuta a zero in modo da far scorrere un periodo di clock. Al clock successivo la rete passa nello stato C, dove l'uscita è uno, e ci resta solo per un periodo di clock, come previsto dal punto 4 della specifica. Lo stato D serve ad attendere il ritorno di x a uno.

Essendo 4 gli stati occorrono 2 variabili di stato, cioè due flip-flop per codificarli. Se si codificano gli stati nel seguente modo: A = 00, B = 01, C = 11, D = 10, si ottengono la tabella delle transizioni di Figura A.85 e le corrispondenti mappe di Karnaugh riportate a fianco.

Dalle mappe di Figura A.85 si ottengono le relazioni seguenti per y_1', y_2' e WAIT:

y_1y_2	x = 0	x = 1
00	01	00
01	11	11
11	10	00
10	10	00

$y'_1y'_2$

y_1y_2	x = 0	x = 1
00	0	0
01	1	1
11	1	0
10	1	0

y'_1

y_1y_2	x = 0	x = 1
00	1	0
01	1	1
11	0	0
10	0	0

y'_2

y_1y_2	
00	0
01	0
11	1
10	0

WAIT

Figura A.85 Tabella delle transizioni e mappe di Karnaugh per le variabili di stato futuro y'_1 e y'_2, e per l'uscita WAIT.

$$y'_1 = y_1\overline{x} + \overline{y_1}y_2; \qquad y'_2 = y_2\overline{y_1} + \overline{y_1x}; \qquad \text{WAIT} = y_1y_2$$

da cui si deduce immediatamente lo schema della rete nel caso di realizzazione con FFD (basta porre $D_1 = y'_1$ e $D_2 = y'_2$).

Se si usano flip-flop JK si ottiene:

$$J_1 = y_2; \qquad K_1 = x; \qquad\qquad J_2 = \overline{y_1x}; \qquad K_2 = y_1$$

La rete risultante è disegnata in Figura A.86. Vale la pena di osservare che se si fosse voluto avere WAIT = 1 per k periodi di clock, sarebbe bastato sostituire lo stato C con una sequenza obbligata di k stati ($\mathrm{C_1, C_2, ..., C_k}$). In questo caso ogni variazione di x nell'intervallo dallo stato B allo stato $\mathrm{C_k}$ compresi sarebbe stato irrilevante. È anche facile ottenere WAIT = 1 dopo m impulsi di clock anziché dopo 1; basta sostituire lo stato B con la sequenza ($\mathrm{B_1, B_2, ...B_m}$).

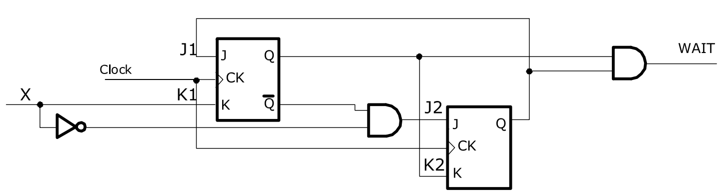


Figura A.86 Generatore di WAIT attraverso una rete sintetizzata col metodo formale.

La rete di Figura A.86 è stata ottenuta adottando un metodo formale per la sua sintesi. Molto spesso il progettista logico usa tecniche e trucchi che gli consentono di arrivare a una soluzione senza passare attraverso tutti i passi del procedimento rigoroso. Nella rete di Figura A.87, ottenuta con ragionamenti di carattere intuitivo, il segnale x viene usato come ingresso di clock al primo flip-flop D (i flip-flop commutano sui fronti di discesa).

Assumiamo che tutti i flip-flop della rete di Figura A.87 siano in stato basso. La situazione non varia fintantoché $x = 1$. Quando x commutata da 1 a 0 causa la variazione da 0 a 1 dell'uscita del primo FF (l'ingresso D di questo flip-flop è posto permanentemente a 1). Al prossimo impulso di clock la stessa variazione si propaga in uscita al secondo

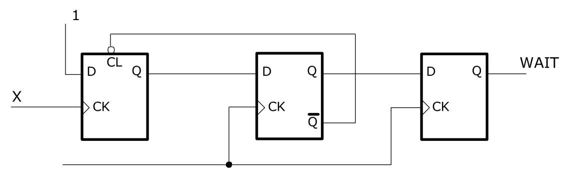


Figura A.87 Generazione del WAIT con una rete costruita con metodo intuitivo.

flip-flop e, contemporaneamente, l'uscita del primo va a 0 per a causa di CL asserito[23]. Dopo un altro impulso di clock anche l'uscita dell'ultimo FF, cioè WAIT, passa a 1 e vi resta solo per un periodo di clock perché l'uscita del secondo FF è intanto tornata a 0.

Si noti che la rete di Figura A.87 riconosce comunque il passaggio basso di x, mentre la rete di Figura A.86 richiede che x debba essere mantenuto basso almeno fino al fronte del clock che lo campiona.

---

[23] Si assume che in presenza di CL asserito il clock non abbia effetto.

Domande ed esercizi

A.1 Si discutano le ragioni che hanno portato all'affermazione dell'elettronica digitale e al suo esclusivo impiego nel mondo dei calcolatori elettronici.

A.2 Costruire la tabella di verità per le seguenti espressioni:
(a) $XYZ + \overline{X}\,\overline{Y}$ (b) $A(\overline{B} + \overline{B}C)$ (c) $(\overline{A}\,\overline{B}(A + C))$ (d) $(X + \overline{Y})(\overline{X} + Z)$

A.3 Sia data una funzione y attraverso le sue forme canoniche sintetiche:
$y = \sum_3(0, 2, 5, 7) = \prod_3(1, 3, 4, 6)$. Si scrivano le corrispondenti espressioni algebriche e si derivi la seconda dalla prima ricorrendo al teorema di De Morgan.

A.4 Semplificare le seguenti espressioni:
(a) $X + \overline{X}\,\overline{Y} + XZ$ (b) $A(\overline{B}C + \overline{B} + \overline{C})$ (c) $\overline{A}\,\overline{B}(A + C)(B + \overline{C})$

A.5 Tracciare per ciascuna delle espressioni dell'Esercizio A.4 le corrispondenti reti, sia per la forma data che per la forma semplificata.

A.6 Si dia una dimostrazione algebrica della validità del teorema di De Morgan per il caso generale di n variabili.

A.7 Si dimostri che qualsiasi espressione può sempre essere ricondotta in forma canonica. Si parta da una funzione espressa in forma algebrica qualunque.

A.8 Scrivere l'espressione minima SP per una rete a tre ingressi w, x, y, la cui uscita z deve essere 1 per le configurazioni di ingresso 010 110 111. Per tutte le altre configurazioni l'uscita deve essere 0.

A.9 Si progetti una rete con due ingressi primari x e y, un'uscita z e tre ingressi di controllo c_1, c_2 e c_3. Non è ammesso che più di un ingresso di controllo sia a 1. Quando $c_1 = 1$, allora $z = x + y$; quando $c_2 = 1$ allora $z = xy$; quando $c_3 = 1$ allora $z = \overline{x}y$.

A.10 Convertire le seguenti espressioni da PS a SP:
(a) $(x + \overline{y})(x + y)$ (b) $(x + y + z)(x + \overline{y} + \overline{z})$ (c) $(x + y)(x + y + z)(\overline{x} + \overline{y} + \overline{z})$

A.11 Per le seguenti espressioni passare alla forma SP, riportarle sulla mappa di Karnaugh e trovare le espressioni minime SP.
(a) $(x + \overline{y}z)(wx + z)(y + z)$ (b) $\overline{xy}(x + w(yz + \overline{yz}))$
(c) $wx\overline{y}z(x + (wx + \overline{z})\overline{y} + z(x + y))$ (d) $(w + x + y + z)(\overline{y} + \overline{z}(x + z))$

A.12 Si consideri la funzione

$$y = x_1x_2(\overline{x}_1x_3 + x_2x_3x_4 + \overline{x}_2\overline{x}_4) + \overline{x}_3x_4(x_1 + x_2 + x_3)$$

Si supponga che per essa ci siano le condizioni di indifferenza corrispondenti all'espressione

$$x_1\overline{x}_4(x_2\overline{x}_3 + x_2x_3) + \overline{x}_1x_2x_3x_4$$

Si trovi l'espressione minima per y usando le mappe di Karnaugh.

A.13 Facendo riferimento al decodificatore BCD del Paragrafo A.2.7 si progetti la rete che segnala se il codice di ingresso non è BCD.

A.14 Si dimostri con trasformazioni algebriche che per l'operatore XOR vale la proprietà associativa; ovvero dimostrare che vale
$x_1 \oplus (x_2 \oplus x_3) == (x_1 \oplus x_2) \oplus x_3 = x_1 \oplus x_2 \oplus x_3$

A.15 Si consideri l'espressione $x_1 + x_2 \cdot (x_3 + x_4 \cdot (x_5 + x_6 \cdot (\ldots)))$, alla quale corrisponde una rete a più di due livelli, detta SPSPSP ...(Somma di Prodotto di Somma di ...). Si trovi una

regola generale per passare alla corrispondente rete di soli NAND. Si tratti lo stesso problema per il caso duale di reti in forma PSPS ...

A.16 Si trasformi in rete di NAND la rete di Figura A.88. Si ripeta l'esercizio trasformandola in rete di NOR.

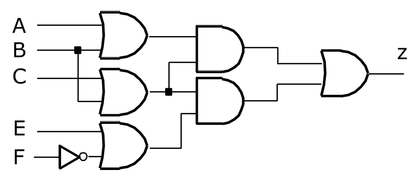


Figura A.88 Rete da trasformare in rete di soli NAND.

A.17 Le reti SP (PS) si trasformano direttamente in reti di NAND (NOR), secondo la tecnica illustrata al Paragrafo A.3.2. Si verifichi che aggiungendo in uscita una ulteriore livello AND (OR), di cui si devono precisare gli ingressi, la trasformazione naturale diventa in reti di NOR (NAND).

A.18 Si dimostri che la rete a destra in Figura A.18 corrisponde all'OR esclusivo.

A.19 Si usi un multiplexer a 8 ingressi per costruire la funzione $A = xy + x(\overline{y} + z)$

A.20 Si usi una memoria PROM 8×2 per costruire le due funzioni

$A = xy + \overline{x}\,\overline{y}z$

$B = \overline{x} + xy + z$

Si adotti la notazione del Paragrafo A.6.6.

A.21 Si discuta l'impiego delle memorie ROM come componente per la generazione di funzioni logiche di molte variabili. In particolare si faccia un confronto con la logica cablata convenzionale.

A.22 Si decodifichi l'indirizzo esadecimale 240 con una rete di AND-OR-NOT.

A.23 Si disegni una rete fatta di sole porte AND/OR che consenta trasferimenti da m registri sorgente a n registri di destinazione. La rete deve permettere tutti i possibili trasferimenti in parallelo, compreso il trasferimento da uno a più registri e il trasferimento dell'OR del contenuto di più sorgenti in una o più destinazioni.

A.24 Si disegni una rete fatta di sole porte AND/OR che realizzi un bus tramite il quale sia possibile il trasferimento del contenuto di una qualunque registro sorgente a un qualunque registri di destinazione.

A.25 Si costruisca un controllore di parità su parole di 4 bit. Il controllore è una rete in cui entrano i quattro bit della parola e ha un'unica uscita il cui valore è 1 nel caso che la parola contenga un numero dispari di bit a 1 e 0 in caso contrario.

A.26 Si discutano i vantaggi/svantaggi della logica sincrona rispetto a quella asincrona.

A.27 In Figura A.43 si sostituiscano le porte NOR con porte NAND e si analizzi il comportamento della rete. In particolare, si verifichi che la configurazione di ingresso esclusa è la 00.

A.28 Partendo dalla rete di Figura A.51 si applichi il metodo del Paragrafo A.3.2 e si ricavi lo schema del latch in termini di sole porte NAND. Si discuta il comportamento della rete.

A.29 Si progettino le reti che costruiscono: (a) un FFSR da un FFT; (b) un FFSR e un FFJK da un FFD; (c) un FFD da un FFT.

A.30 Il contenuto di un registro di 8 bit deve assumere il valore F0 (esadecimale) al momento della messa sotto tensione e quando viene asserito il comando R (*reset*). Si costruisca la corrispondente rete di comando ricorrendo all'impiego degli ingressi asincroni PR e CL (Paragrafo A.8.2).

A.31 Analizzare il comportamento della rete di Figura A.89. Si supponga di partire dallo stato 00.

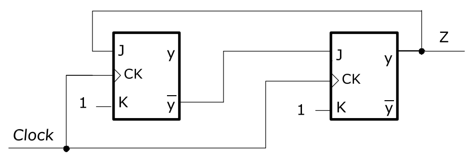


Figura A.89 Rete per l'Esercizio A.31.

A.32 Dato il diagramma di stato a destra in Figura A.89, progettare la rete corrispondente.

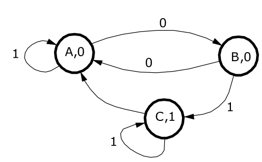


Figura A.90 Diagramma di stato per l'Esercizio A.32.

A.33 La rete di Figura A.91 ha la funzione di riconoscitore di una sequenza di ingresso. Quale?

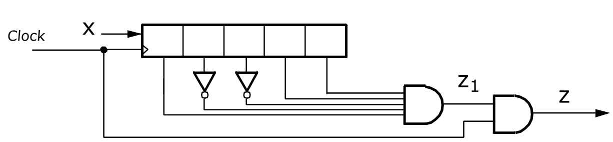


Figura A.91 Il blocco a cui viene portato il segnale di ingresso x è un registro a scorrimento.

A.34 Nel caso della trasmissione seriale di caratteri di n bit, il calcolo della parità può essere effettuato, in trasmissione e in ricezione, da una rete sequenziale che calcoli il bit di parità sulla stringa di bit via via trasmessi (ricevuti). Si progetti una simile rete per $n = 4$ (in trasmissione il bit di parità calcolato deve essere trasmesso come quinto bit).

A.35 Si progetti un contatore asincrono modulo 2^n e lo si realizzi con flip-flop D o con flip-flop JK.

A.36 Si progetti un contatore sincrono modulo $N \neq 2^n$ e lo si realizzi con flip-flop D o con flip-flop JK.

A.37 Si costruisca la logica che permette a un contatore ad anello di contare fino a 5 o fino a 7 a seconda dello stato del segnale di controllo X.

A.38 Partendo da un contatore modulo 2^n si progetti la logica che a seconda del valore del segnale di controllo UD (*up/down*) conta in aumento o in diminuzione.

A.39 Con riferimento all'Esercizio A.38, si aggiunga la logica di precaricamento di un contatore verso il basso e si blocchi il conteggio quando il contenuto del contatore diventa 0. Il precaricamento deve avvenire sul clock che campiona a 1 il segnale LD (*load*). Dopo che LD è stato trovato a 1 esso non ha più effetto fino al clock successivo a quello che porta a 0 il contatore.

Soluzioni degli esercizi appendice A

Aggiornato il 31 marzo 2017

A.2 L'esercizio viene svolto per le espressioni (a) e (b).

X	Y	Z	XYZ	$\overline{X}\,\overline{Y}$	$XYZ+\overline{X}\,\overline{Y}$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	0	1

Tabella A.1 Tabella di verità della funzione (a) (Esercizio A.2).

A	$\overline{A}$	$\overline{B}$	C	$A+C$	$\overline{A}\,\overline{B}$	$\overline{A}\,\overline{B}(A+C)$
0	1	1	0	0	1	0
0	1	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	0	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0

Tabella A.2 Tabella di verità della funzione (b) (Esercizio A.2). Sono state riportate le variabili nella forma con cui entrano nell'espressione.

A.4 L'espressione $X+\overline{X}\,\overline{Y}+XZ$ si semplifica applicando al proprieta di assorbimento.

$$\begin{aligned} X+\overline{X}\,\overline{Y}+XZ &= X(1+Z)+\overline{X}\,\overline{Y} \quad = X+\overline{X}\,\overline{Y} \quad = \\ &= (X+\overline{X})(X+\overline{Y}) \quad = 1(X+\overline{Y}) \\ &= X+\overline{Y} \end{aligned}$$

L'espressione $A(\overline{B}\,C+\overline{B}+\overline{C})$ si semplifica applicando la proprietà di assorbimento e successivamente la proprietà distributiva.

$$A(\overline{B}C+\overline{B}+\overline{C}) = A(\overline{B}+\overline{B}+\overline{C}) = A(\overline{B}+\overline{C}) = A\overline{B}+A\overline{C}$$

L'espressione $\overline{A}\,\overline{B}(A+C)(B+\overline{C})$ si semplifica nel modo seguente

$$\overline{A}\,\overline{B}(A+C)(B+\overline{C}) = (\overline{A}\,\overline{B}A+\overline{A}\,\overline{B}C)(B+C) = \overline{A}\,\overline{B}C(B+C) = \overline{A}\,\overline{B}C$$

A.5 La rete corrispondente alla funzione (a) è mostrata a sinistra in Figura A.92, mentre la semplificazione è a destra. La rete corrispondente alla funzione (b) è mostrata a sinistra in Figura A.92, mentre la semplificazione a destra.

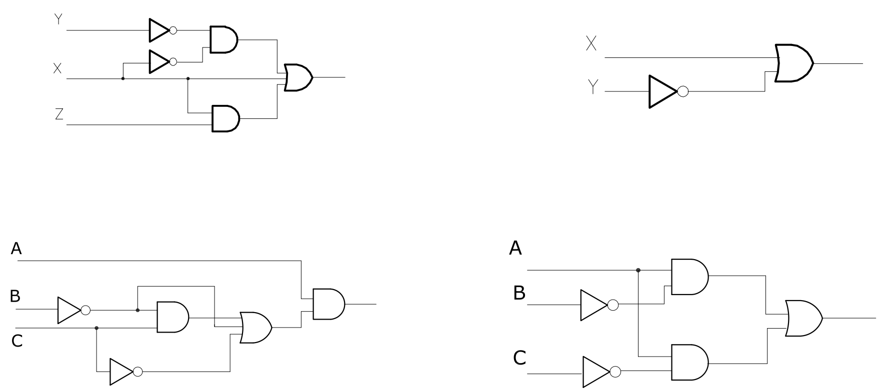


Figura A.92 (Esercizio A.5) Fascia alta: rete originale e semplificata del punto (a). Fascia bassa: rete originale e semplificata del punto (b).

A.6 Si vuole arrivare a scrivere l'uguaglianza di De Morgan nella forma:

$$\overline{x_1 + x_2 + \ldots + x_n} = \overline{x_1} \cdot \overline{x_2} \cdot \ldots \cdot \overline{x_n}$$

Posto $x_2 + x_3 + \ldots + x_n = X$, sostituendo nella precedente e applicando il teorema di De Morgan per due variabili si ha:

$$\overline{x_1 + x_2 + \ldots + x_n} = \overline{x_1 + X} = \overline{x_1} \cdot \overline{X} = \overline{x_1} \cdot (\overline{x_2 + \ldots + x_n}).$$

Applicando iterativamente la semplificazione si ottiene l'uguaglianza richiesta.

A.7 Si faccia riferimento alla prima forma canonica. Data la generica funzione $f(x_1, x_2, \ldots, x_n)$

- Se $f(x_1, x_2, \ldots, x_n)$ contiene generici termini di somme e prodotti, è sempre possibile riportare f ad una somma di prodotti svolgendo le operazioni algebriche (i prodotti) in f.
- Quando f è in forma di somma di prodotti è sempre possibile riportare i singoli prodotti a mintermini, moltiplicandoli con termini in forma $(x_i + \overline{x_i})$, x_i essendo una variabile che non compare nel prodotto.

Esempio: sia data $f(x_1, x_2, \ldots, x_n) = x_1 \cdot (x_2 + x_3) + x_2\overline{x_3}$.

$$\begin{aligned}
x_1 \cdot (x_2 + x_3) + x_2\overline{x_3} &= x_1x_2 + x_1x_3 + x_1\overline{x_3} \cdot (x_2 + \overline{x_2}) = \\
&= x_1x_2 \cdot (x_3 + \overline{x_3}) + x_1x_3 \cdot (x_2 + \overline{x_2}) + x_1x_2\overline{x_3} + x_1\overline{x_2}\,\overline{x_3} = \\
&= x_1x_2x_3 + x_1x_2\overline{x_3} + x_1x_2x_3 + x_1\overline{x_2}x_3 + x_1x_2\overline{x_3} + x_1\overline{x_2}\,\overline{x_3} = \\
&= x_1x_2x_3 + x_1x_2\overline{x_3} + x_1\overline{x_2}x_3 + x_1\overline{x_2}\,\overline{x_3}.
\end{aligned}$$

A.8 La funzione di uscita della rete può essere scritta come $f = \overline{w}x\overline{y} + wx\overline{y} + wxy$. Questa si semplifica nel modo che segue.

$$f = \overline{w}x\overline{y} + wx\overline{y} + wxy = \overline{w}x\overline{y} + wx\overline{y} + wx\overline{y} + wxy = x\overline{y} + wx = x(w + \overline{y})$$

A.9 L'espressione di z si ricava direttamente dalla specifica prendendo la parola "allora" come OR e la parola "quando" come AND.

$$z = (x + y)c_1 + xyc_2 + \overline{x}yc_3$$

Con le mappe si può verificare che questa è l'espressione minima.

A.10 Per il caso (a) vale quanto segue.

$$(x + \overline{y})(x + y) = \; xx + xy + x\overline{y} + y\overline{y} = \; x + x(y + \overline{y}) + 0 = \; x + x = \; x$$

A.11

- Per la funzione (a) si ha: $F = (x + \overline{y}z)(wx + z)(y + z) = wy + wz + xzy + xz + xwz + x\overline{y}wz + \overline{y}z$.
  La mappa di F è in Figura A.93. La copertura di F fornisce l'espressione minima: $F = \overline{y}z + wy + wz + xz$.

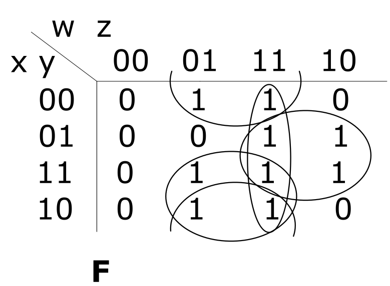


Figura A.93 Mappa della funzione (a) dell' Esercizio A.11.

- Per la funzione (b) si ha: $F = \overline{xy}(x + w(yz + \overline{yz})) = (\overline{x} + \overline{y})(x + w(yz + \overline{y} + \overline{z})) = (\overline{x}+\overline{y})(x+wyz+w\overline{y}+w\overline{z}) = \overline{x}ywz+\overline{x}w\overline{y}+\overline{x}w\overline{z}+\overline{x}y+wz+\overline{y}+\overline{y}w\overline{z} = wz+\overline{y}+\overline{x}y+\overline{x}w\overline{z}$.
  Riportando F sulla mappa e coprendo come in Figura A.94 si ottiene la funzione $F = \overline{y} + x + wz$.
- Per la funzione (c) si ha: $F = wx\overline{y}z(x+(wx+\overline{z})\overline{y}+z(x+y)) = wx\overline{y}z(x+(wx\overline{y}+\overline{z}\,\overline{y})+zx+zy) = wx\overline{y}z(x+wx\overline{y}+\overline{z}\,\overline{y}+zx+zy) = wx\overline{y}zx+wx\overline{y}zwx\overline{y}+wx\overline{y}z\overline{z}\,\overline{y}+wx\overline{y}zzx+wx\overline{y}zzy = wx\overline{y}zx$.
  Non è quindi necessario procedere alla creazione della tabella di verità e della mappa in quanto la funzione è già ridotta al minimo.

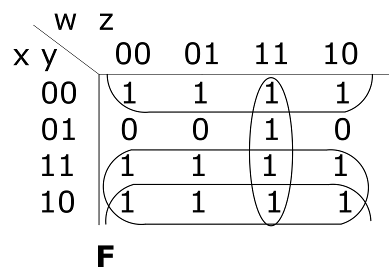


Figura A.94 Mappa di della funzione (b) dell' Esercizio A.11.

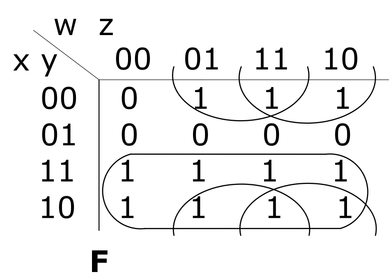


Figura A.95 Mappa della funzione (d) dell'Esercizio A.11.

- Per la funzione (d) si ha: $F = (w+x+y+z)(\overline{y}+\overline{z}x) = w\overline{y}+w\overline{z}x+x\overline{y}+\overline{z}x+xy\overline{z}+z\overline{y}+x$ $= x + z\overline{y} + w\overline{y}$. E' facile verificare che l'espressione non è ulteriormente semplificabile. Del resto per essa vale la mappa di Karnaugh di Figura A.95 dalla quale si ottiene $F = x + z\overline{y} + w\overline{y}$.

A.12 Prima di tutto si riportano le espressioni in forma SP.

$$\begin{aligned} y = \; & x_1x_2(\overline{x}_1x_3 + x_2x_3x_4 + \overline{x}_2\overline{x}_4) + \overline{x}_3x_4(x_1 + x_2 + x_3) = \\ = \; & x_1x_2\overline{x}_1x_3 + x_1x_2x_2x_3x_4 + x_1x_2\overline{x}_2\overline{x}_4 + \overline{x}_3x_4x_1 + \overline{x}_3x_4x_2 + \overline{x}_3x_4x_3 = \\ = \; & 0 + x_1x_2x_3x_4 + 0 + \overline{x}_3x_4x_1 + \overline{x}_3x_4x_2 + 0 = \\ = \; & x_1x_2x_3x_4 + x_1\overline{x}_3x_4 + x_2\overline{x}_3x_4 \end{aligned}$$

Le condizioni di indifferenza si esprimono come segue

$$d = \; x_1\overline{x}_4(x_2\overline{x}_3 + x_2x_3) + \overline{x}_1x_2x_3x_4 = \; x_1x_2\overline{x}_3\overline{x}_4 + x_1x_2x_3\overline{x}_4 + \overline{x}_1x_2x_3x_4$$

Si disegna quindi la mappa, riportando gli 1 della funzione e le condizioni di indifferenza. Con la copertura di Figura A.96 si ottiene $f = x_2x_4 + x_1\overline{x}_3x_4$

A.13 In un decodificatore BCD si hanno 6 condizioni di indifferenza, come si può notare dalla Tabella di Figura A.11 del testo. Si tratta quindi di coprire le caselle della mappa che sono condizioni di indifferenza. E' facile verificare che si ottiene: $Z = AB+AC = A(B+C)$

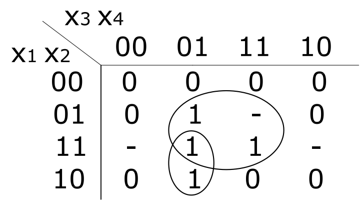


Figura A.96 Mappa per l'Esercizio A.12.

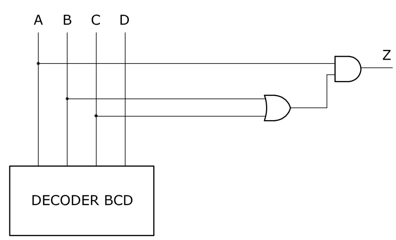


Figura A.97 Rete dell'Esercizio A.13.

dove Z è il segnale che indica la presenza di codice non BCD. La Figura A.97 mostra la rete conseguente.

A.14 La dimostrazione può essere fatta nel seguente modo:

$$\begin{aligned}
x_1 \oplus (x_2 \oplus x_3) &= x_1(\overline{x_2 \oplus x_3}) + \overline{x_1}(x_2 \oplus x_3) \\
&= x_1(\overline{x_2\overline{x_3} + x_3\overline{x_2}}) + \overline{x_1}(\overline{x_2}x_3 + \overline{x_3}x_2) \\
&= x_1(\overline{x_2\overline{x_3}}\,\overline{x_3\overline{x_2}}) + \overline{x_1}\,\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3} \\
&= x_1(\overline{x_2} + x_3)(\overline{x_3} + x_2) + \overline{x_1}\,\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3} \\
&= x_1(x_2x_3 + \overline{x_2}\,\overline{x_3}) + \overline{x_1}\,\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3} \\
&= x_1x_2x_3 + x_1\overline{x_2}\,\overline{x_3} + \overline{x_1}\,\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3}
\end{aligned}$$

Mentre:

$$\begin{aligned}
(x_1 \oplus x_2) \oplus x_3 &= (x_1 \oplus x_2)\overline{x_3} + x_3(\overline{x_1 \oplus x_2}) \\
&= (x_1\overline{x_2} + \overline{x_1}x_2)\overline{x_3} + x_3(\overline{x_2\overline{x}_1 + x_1\overline{x_2}}) \\
&= x_1\overline{x_2}\,\overline{x_3} + \overline{x_1}x_2\overline{x_3} + x_3(\overline{x_1\overline{x_2}}\,\overline{x_2\overline{x_1}}) \\
&= x_1\overline{x_2}\,\overline{x_3} + \overline{x_1}x_2\overline{x_3} + x_3(\overline{x_1} + x_2)(\overline{x_2} + x_1) \\
&= x_1\overline{x_2}\,\overline{x_3} + \overline{x_1}x_2\overline{x_3} + x_3(x_1x_2 + \overline{x_1}\,\overline{x_3}) \\
&= x_1\overline{x_2}\,\overline{x_3} + \overline{x_1}x_2\overline{x_3} + x_1x_2x_3 + \overline{x_1}\,\overline{x_2}x_3
\end{aligned}$$

Dunque $x_1 \oplus (x_2 \oplus x_3) = (x_1 \oplus x_2) \oplus x_3$, e pertanto vale la proprietà associativa. Ne consegue $x_1 \oplus (x_2 \oplus x_3) = (x_1 \oplus x_2) \oplus x_3 = x_1 \oplus x_2 \oplus x_3$.

A.15 Data l'espressione $x_1 + x_2 \cdot (x_3 + x_4 \cdot (x_5 + x_6 \cdot (\ldots)))$ complementando due volte e applicando il teorema di De Morgan si ha:

$$\begin{aligned}
x_1 + x_2 \cdot (x_3 + x_4 \cdot (x_5 + x_6 \cdot (\ldots))) &= \overline{\overline{x_1 + x_2 \cdot (x_3 + x_4 \cdot (x_5 + x_6))}} = \\
&= \overline{\overline{x_1} \cdot \overline{x_2 \cdot (x_3 + x_4 \cdot (x_5 + x_6))}} = \\
&= \overline{x_1} \mid \overline{x_2 \cdot (x_3 + x_4 \cdot (x_5 + x_6))} = \\
&= \overline{x_1} \mid (x_2 \mid (x_3 + x_4 \cdot (x_5 + x_6))) = \\
&= \overline{x_1} \mid (x_2 \mid \overline{\overline{(x_3 + x_4 \cdot (x_5 + x_6))}}) = \\
&= \overline{x_1} \mid (x_2 \mid \overline{\overline{x_3} \cdot \overline{(x_4 \cdot (x_5 + x_6))}}) = \\
&= \overline{x_1} \mid (x_2 \mid \overline{x_3} \mid \overline{(x_4 \cdot (x_5 + x_6))}) = \\
&= \overline{x_1} \mid (x_2 \mid \overline{x_3} \mid x_4 \mid (x_5 + x_6)) = \\
&= \overline{x_1} \mid (x_2 \mid \overline{x_3} \mid x_4 \mid \overline{x_5} \mid \overline{x_6}).
\end{aligned}$$

La regola è che una rete di SPSPSP... si trasforma in una rete di soli NAND sostituendo tutte le porte della rete data con porte NAND e complementando (ovvero passandoli attraverso porte NAND) gli ingressi di ordine dispari nell'espressione (il primo termine della SPSPSP... ha indice 1). Si veda la Figura A.98. In modo analogo si prova che le reti PSPSPS... si trasformano in reti di soli NOR.

$$\begin{aligned}
x_1.(x_2 + x_3 \cdot (x_4 + x_5 \cdot (x_6(\ldots))) &= \overline{\overline{x_1.(x_2 + x_3 \cdot (x_4 + x_5 \cdot (x_6(\ldots)))}} = \\
&= \overline{\overline{x_1} + \overline{x_2 + (x_3(x_4 + (x_5 \cdot (x_6))}} = \textit{eccetera} \\
&= \overline{x_1} \mid \overline{x_2 \cdot (x_3 + x_4 \cdot (x_5 + x_6))} = \\
&= \overline{x_1} \mid (x_2 \mid (x_3 + x_4 \cdot (x_5 + x_6))) = \\
&= \overline{x_1} \mid (x_2 \mid \overline{\overline{(x_3 + x_4 \cdot (x_5 + x_6))}}) = \\
&= \overline{x_1} \mid (x_2 \mid \overline{\overline{x_3} \cdot \overline{(x_4 \cdot (x_5 + x_6))}}) = \\
&= \overline{x_1} \mid (x_2 \mid \overline{x_3} \mid \overline{(x_4 \cdot (x_5 + x_6))}) = \\
&= \overline{x_1} \mid (x_2 \mid \overline{x_3} \mid x_4 \mid (x_5 + x_6)) = \\
&= \overline{x_1} \mid (x_2 \mid \overline{x_3} \mid x_4 \mid \overline{x_5} \mid \overline{x_6}).
\end{aligned}$$

A.16 La trasformazione si ottiene con la "tecnica dei pallini" del Paragrafo A.3.2. A partire dalla rete originale si introducono i pallini di negazione come a sinistra di Figura A.99. Ciò richiede l'introduzione dei negatori degli ingressi A, B, C, D, E. La rete a sinistra di Figura A.99 si ridisegna come a destra, usando l'usuale simbolo del NAND. In altre parole si tratta di sostituire tutte le porte con porte NAND e di negare gli ingressi di livello dispari.

La Figura A.100 mostra a sinistra l'applicazione dei pallini e a destra la rete risultate in forma di soli NOR.

A.17 L'aggiunta sull'uscita di una porta AND (OR) in una rete SP (PS) trasforma la rete SP (PS) in una PSP (SPS). Le reti SPS si trasformano in reti di NAND e le reti PSP in reti di NOR. Si veda l'Esercizio A.16.

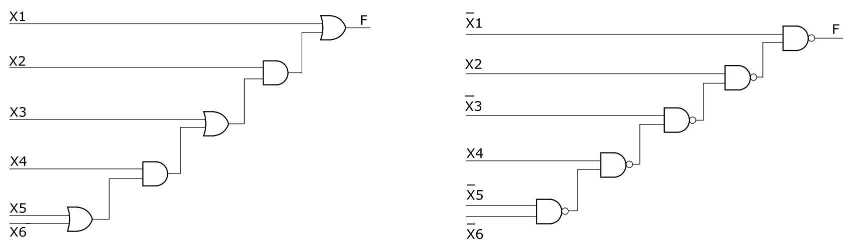


Figura A.98 A sinistra è mostrata la rete originaria dell'Esercizio A.15, mentre a destra la sua trasformazione in sole porte NAND.

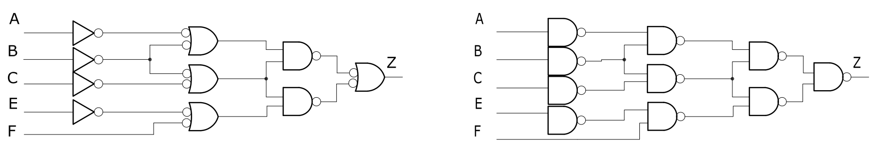


Figura A.99 (Esercizio A.16) Successione dei passaggi per arrivare alla rete equivalente in forma di sole porte NAND.

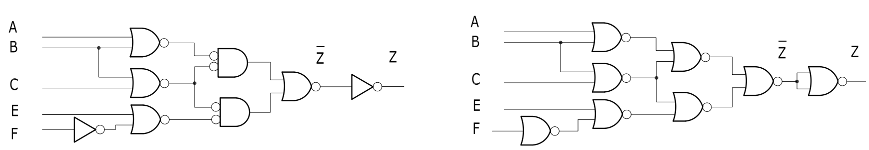


Figura A.100 (Esercizio A.16) Successione dei passaggi per arrivare alla rete equivalente in forma di sole porte NOR.

A.18 Applicando il teorema di De Morgan si ottiene che la rete a destra in Figura A.18 del testo rappresenta la funzione logica: $F(A,B) = (A|(A|B))|(B|(A|B))\overline{\overline{\overline{A}B}\,\overline{A\overline{B}}}$.

$$
\begin{aligned}
\overline{\overline{\overline{A}B}\,\overline{A\overline{B}}} &= \overline{(A+\overline{B})(\overline{A}+B)} \\
&= \overline{A\overline{A}+AB+\overline{A}\,\overline{B}+B\overline{B}} \\
&= \overline{AB+\overline{A}\,\overline{B}} \\
&= (\overline{A}+\overline{B})(A+B) \\
&= \overline{A}B+\overline{B}A \\
&= A\oplus B.
\end{aligned}
$$

A.19 Prima di tutto si riporta la funzione alla forma canonica attraverso espansioni e successive semplificazioni. Si ottiene:

$A = xy + x(\overline{y} + z) = xy + x\overline{y} + xz) = xyz + xy\overline{z} + x\overline{y}z + x\overline{yz} + xyz + x\overline{yz}$ Ovvero $A = x\overline{yz} + x\overline{y}z + xy\overline{z} + xyz$, cioè $f(x, y, z) = \sum(4, 5, 6, 7)$. Dunque le linee 4, 5, 6, 7 del mux vengono fissate a 1 mentre le rimanenti vengono impostate a 0, come in Figura A.19.

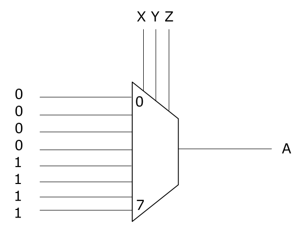


Figura A.101 Rete dell'Esercizio A.19

A.20 Anzitutto occorre dare la forma canonica delle due funzioni di uscita.

$A = xy + \overline{x}\,\overline{y}z = xyz + xy\overline{z} + \overline{x}\,\overline{y}z = \sum(1, 6, 7)$

$$
\begin{aligned}
\text{B} \quad = \quad \overline{x} + xy + z = \quad &= \overline{x}y + \overline{x}\,\overline{y} + xyz + xy\overline{z} + xz + \overline{x}z \\
&= \overline{x}yz + \overline{x}y\overline{z} + \overline{x}\,\overline{y}z + \overline{x}\,\overline{y}\,\overline{z} + xy\overline{z} + xyz + x\overline{y}z + \overline{x}yz\overline{x}\,\overline{x}z \\
&= \overline{x}\,\overline{y}z + \overline{x}\,\overline{y}\,\overline{z} + \overline{x}y\overline{z} + \overline{x}yz + x\overline{y}z + xy\overline{z} + xyz \\
&\quad\;\; 1 \qquad 0 \qquad 2 \qquad\quad 3 \qquad 5 \qquad\quad 6 \qquad 7 \\
&= \sum(0, 1, 2, 3, 5, 6, 7).
\end{aligned}
$$

Le due espressioni permettono di fissare i collegamenti interni della PROM di Figura A.102, (in essa il collegamento è rappresentato da una $\times$ cerchiata).

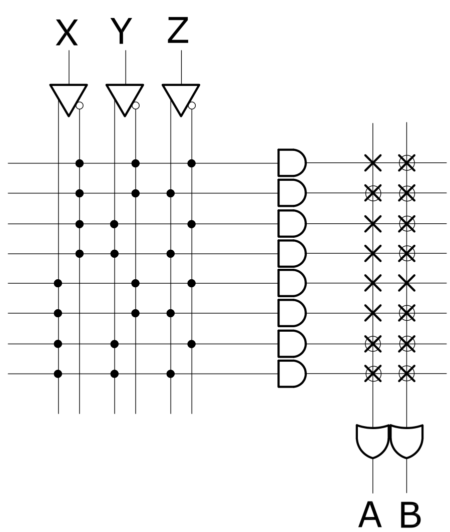


Figura A.102 PROM 8x2 che realizza le funzioni $A = xy + \overline{y}x\overline{z}$ e $B = \overline{x} + xy + z$ (Esercizio A.20).

A.22 Il numero esadecimale 240 corrisponde in binario al numero 0010 0100 0000.

Assumendo pari a 4 il fan-in delle porte AND si può predisporre una rete di decodifica per ogni parola di 4 bit. Ovviamente dispondendo di una porta AND con fan-in pari a 12 ne basta una sola. L'uscita Z viene abilitata solo se gli ingressi corrispondono alla codifica binaria di 240. La rete è mostrata in Figura A.103.

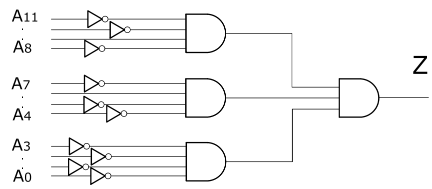
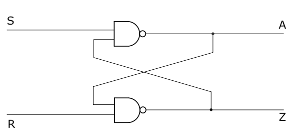


Figura A.103 Rete di decodifica indirizzo 240h(Esercizio A.22).

A.25 La risposta è banale: basta una porta XOR a 4 ingressi.

A.27 La rete di Figura A.43 del testo è un latch realizzato con porte NOR. Sostituendo le porte NOR con porte NAND si ottiene la rete di Figura A.27. Il latch di NAND ha un comportamento duale rispetto a quello di NOR. Ovvero, la configurazione di ingresso non consentita è 00, quella di riposo 11.



S	R	A	Z
1	1	$\overline{Z}$	$\overline{A}$
1	0	1	0
0	1	0	1
0	0	1	1

Figura A.104 Rete dell'Esercizio A.27 e suo comportamento.

A.28 Seguendo la "tecnica dei pallini" (Paragrafo A.3.2), dalla Figura A.43 riportata in alto a sinistra in Figura A.105, si ottiene la rete in alto a destra di Figura A.105. Successivamente, si esegue la sostituzione con porte NAND, ottenendo il latch in basso a sinistra di Figura A.105, la cui interpretazione in logica negativa è mostrata a destra. La differenza tra latch di NOR e quello di NAND è la stessa che intercorre tra logica positiva e negativa. Ovvero le reti hanno lo stesso funzionamento a patto di considerare gli 0 come 1 e viceversa (si veda anche l'Esercizio A.27).

A.29 In Figura A.106 viene presentato lo schema generale di soluzione del problema. Si tratta di definire la rete combinatoria che nel caso (a) ha come ingressi S, R e y, e la cui uscita T ha l'effetto di far commutare il flip flip in modo che appaia come un SR. Questo schema, mutatis mutandis, si usa per qualunque trasformazione tra diversi tipi di flip flop.

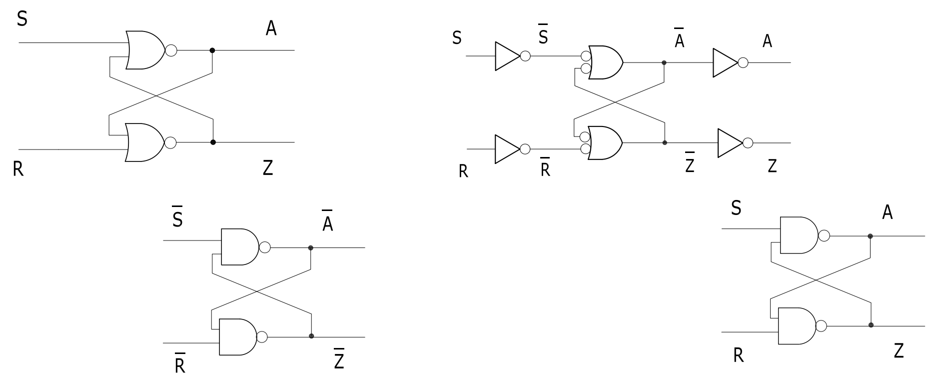


Figura A.105 (Esercizio A.28) Sopra a sinistra il latch di NOR di Figura A.43, a destra la sua trasformazione con la "tecnica dei pallini". Sotto a sinistra passaggio a soli NAND, a destra l'interpretazione in logica negativa.

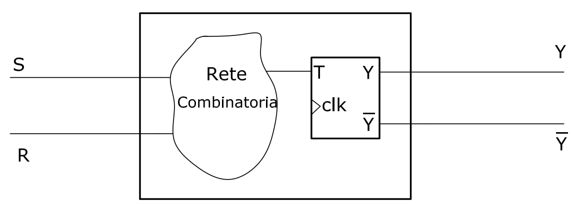


Figura A.106 (Esercizio A.29) Schematizzazione del problema della creazione di un FFSR sfruttando un FFT. Lo schema è valido anche nel caso generale.

Il problema del caso (a) è schematizzato in Figura A.106. In Figura A.107, a sinistra, sono mostrati gli ingressi richiesti per le possibili transizioni dei due tipi di FF. La mappa di y' in funzione di S, R e y è al centro. La mappa di T assume la forma a destra in Figura A.107, la cui copertura porta alla seguente espressione per T:

$$T = S\overline{y} + Ry$$

La rete corrispondente è riportata in Figura A.108.

Il punto (b) chiede di ricavare un FFSR da un FFD e un FFJK da un FFD. In Tabella A.3 sono mostrate le 3 tabelle di transizione per i FF usati. Nel caso del FFD non c'è bisogno di passare attraverso le mappe. Infatti, poichè per un FFD vale l'equazione di stato $y' = D$, si tratta semplicemente di imporre $D = S + \overline{R}y$ per il caso della trasformazione FFSR → FFD e $D = J\overline{y} + \overline{K}y$ per la trasformazione FFJK → FFD. Si trovano così le reti mostrate rispettivamente a sinistra e destra in Figura A.109.

Si lascia al lettore la costruzione di un FFD da un FFT.

y → y’	S	R	T
0 → 0	0	-	0
0 → 1	1	0	1
1 → 0	0	1	1
1 → 1	-	0	0

SR \ y	0	1
00	0	1
01	0	0
11	-	-
10	1	1

y'

SR \ y	0	1
00	0	0
01	0	1
11	-	-
10	1	0

T

Figura A.107 (Esercizio A.29caso a). Transizioni di stato e corrispondenti ingressi per i Flip Flop SR e T e mappe corrispondenti.

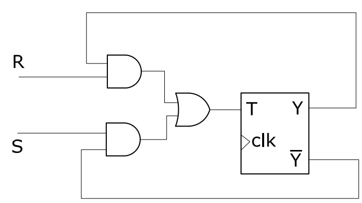


Figura A.108 Rete Esercizio A.29 caso a.

y → y’	S	R	J	K	D
0 → 0	0	-	0	0	0
0 → 1	1	0	1	0	1
1 → 0	0	1	0	1	0
1 → 1	-	0	0	0	1

Tabella A.3 (Esercizio A.29) Transizioni di stato e corrispondenti ingressi per i FFSR, FFJK e FFD

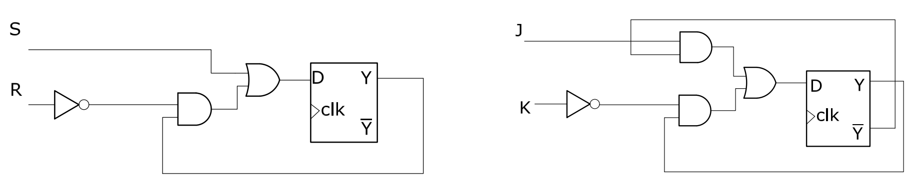


Figura A.109 (Esercizio A.29) Reti che realizzano, a sinistra un FFSR a partire da un FFD, mentre a destra un FFJK da un FFD.

A.30 Il valore F0h corrisponde a 1111 0000b. Qualunque sia il tipo di FF utilizzato per realizzare il registro, si tratta di legare i segnali asicroni Clear e Preset CL a massa o a Vcc in modo da portare i singoli FF al valore richiesto all’atto della messa sotto tensione. La rete a di Figura A.110 genera il segnale Clear/Preset (assumendo che debbano essere attivi bassi) quando viene asserito R. Si noti la presenza di un inverter *Schmitt triggered*, per generare un’onda il più possibile quadra.

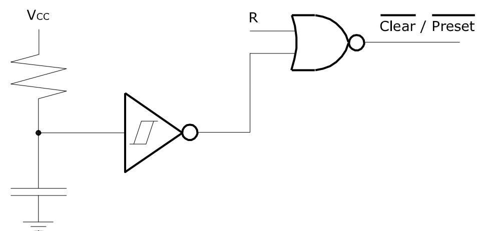


Figura A.110 (Esercizio A.30) Rete che genera il segnale Clear/Preset.

A.31 Tenuto conto del funzionamento del FFJK (a sinistra in Tabella A.89), la tabella delle transizioni dell rete di Figura A.89, a partire dallo stato 00 è quello a destra in Tabella A.4. La rete è quindi un contatore modulo 3.

J	K	y'
0	0	y
0	1	0
1	0	1
1	1	$\overline{y}$

Clock n°	$y_0\,y_1$	$y'_0\,y'_1$
0	00	01
1	01	10
2	10	00

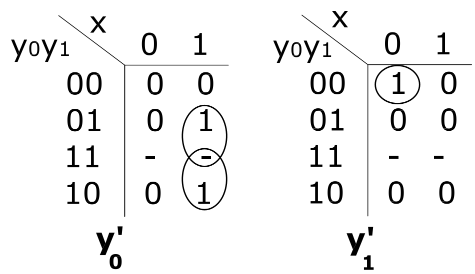
Tabella A.4 (Esercizio A.31) A sinistra la tabella del FFJK, a destra la tabella delle transizioni della rete Figura A.89. Con y_0 e y_1 si indicano rispettivamente l'uscita del FF di sinistra e di destra di Figura A.89.

A.32 Utilizzando per gli stati A,B,C la seguente codifica: A 00, B 01 e C 10, si ottiene la tabella delle transizioni mostrata in Figura A.111.

$y_0\,y_1$ \ x	0	1	
00	01	00	0
01	00	10	0
11	-	-	-
10	00	10	1
	$y'_0\,y'_1$		z

Figura A.111 (Esercizio A.32) Tabella delle transizioni.

Alla tabella delle transizioni corrispondono le mappe a sinistra in Figura A.112 dalle quali si ricavano le funzioni a destra nella stessa Figura. Infine la realizzazione della rete è mostrata in Figura A.113.



$$y_0' = y_0 x + y_1 x$$
$$y_1' = \overline{y_0}\,\overline{y_1}\,\overline{x}$$
$$z = y_0$$

Figura A.112 Mappe e funzioni della Tabella A.111 dell'Esercizio A.32.

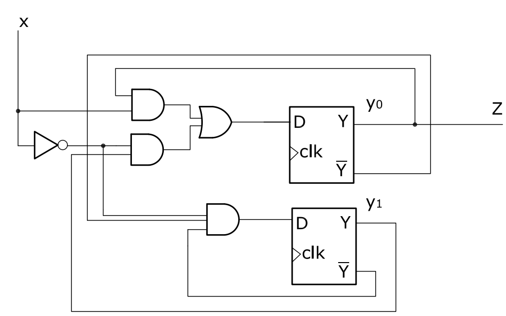


Figura A.113 Rete logica dell' Esercizio A.32.

A.34 La rete che deve essere progettata, ha un unico ingresso (x) e una sola uscita (z) che al 5° clock presenta la parità, mentre sui precedenti presenta x. In Figura A.114 è mostrato lo schema in questione.

In Figura A.115 c'è il diagramma di transizione degli stati. La notazione usata per etichettare gli archi è quella relativa al modello di Mealy. La colonna a sinistra (stati senza apici) individua un numero pari di 1, quella a destra un numero dispari. Supponendo di seguire lo schema a "parità pari", sul 5° clock si ha z=0 se si è nello stato 4, altrimenti z=1 se si è nello stato 4', (indipendentemente da x).

Si può ora ricavare la tabella di flusso e delle transizioni di stato (Tabella A.5).

Codificando gli stati come qui di seguito si ottengono le mappe in Figura A.116.

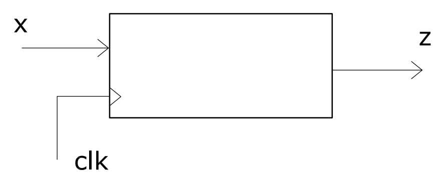


Figura A.114 Schema per l' Esercizio A.34.

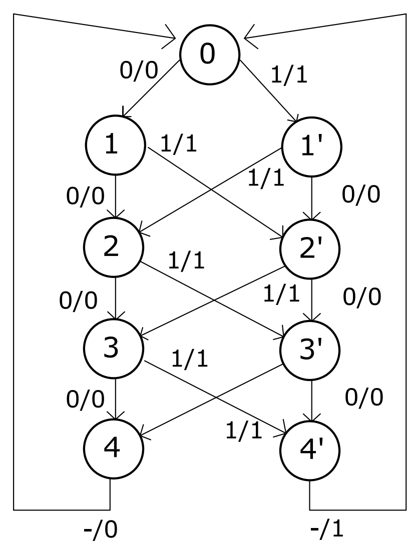


Figura A.115 (Esercizio A.34) Diagramma degli stati.

	x	
SP	0	1
0	1,0	1',1
1	2,0	2',1
2	3,0	3',1
3	4,0	4',1
4	0,0	0,0
1'	2',0	2,1
2'	3',0	3,1
3'	4',0	4,1
4'	0,1	0,1
	SF,z	

Tabella A.5 Tabella di flusso corrispondente al diagramma di stato di Figura A.115

0:0000	
1:0001	1’:1001
2:0010	2’:1010
3:0011	3’:1011
4:0100	4’:1100

Si noti che questa non è la codifica più conveniente in termini di minimizzazione della rete, ma è quella più naturale (rende comprensibile lo stato). Dalle mappe si ricavano le seguenti funzioni di stato e uscita, dalle quali può essere dedotto lo schema della rete.

$y_0' = \overline{y_1}(y_0 \oplus x)$
$y_1' = y_2(\overline{y_3} + y_0)$
$y_2' = \overline{y_0}y_3 + y_0\overline{y_1}\,\overline{y_2}$
$y_3' = y_3(\overline{y_0}y_2 + y_0\overline{y_2}) + \overline{y_0}\,\overline{y_1}\,\overline{y_2}\,\overline{y_3}$
$z = y_0y_1\overline{y_2}\,\overline{y_3}\,\overline{x} + x(\overline{y_0} + \overline{y_1})$.

X=0

y_2y_3 \ y_0y_1	00	01	11	10
00	0001,0	0000,0	0000,1	1010,0
01	0010,0			1011,0
11	0011,0			1100,0
10	0100,0			

y_0' y_1' y_2' y_3' ,z

X=1

y_2y_3 \ y_0y_1	00	01	11	10
00	1001,1	0000,1	0000,0	0010,1
01	1010,1			0011,1
11	1011,1			0100,1
10	1100,1			

y_0' y_1' y_2' y_3' ,z

Figura A.116 Mappe che rappresentano la Tabella A.5 secondo la codifica indicata.Esercizio 46.

Soluzione alternativa

La rete appena vista è ridondante, in realtà bastano due stati per ricordare se il numero di 1 passati è dispari o pari. Possiamo scomporre il problema in modo da avere

- Un contatore che fornisce il segnale c sul 5° clock
- Una rete (due soli stati) che calcola p (parità)
- Una rete che presenta 0 su z sui clock da 0 a 3, e p clock 4

Saprebbe il lettore progettare tale rete?

A.33 La sequenza riconosciuta è 10011. Z_1=1 solo dopo il fronte di clock che ha fatto caricare nel registro questa configurazione (il quinto fronte). Se i FF commutano sul fronte di salita, Z è 1 durante il $\triangle_1$ del clock che ha caricato 10011 mentre se i flip flop commutano sul fronte di discesa, Z è 1 durante $\triangle_2$ del medesimo clock.

A.35 Cominciamo con la realizzazione tramite FFJK. Per semplicità consideriamo il caso $n = 2$. Si tratta di collegare i flip flop in modo tale che ciascun FF divida la frequenza del clock per 2 e di usare lo stato del FF come clock per il successivo. Con

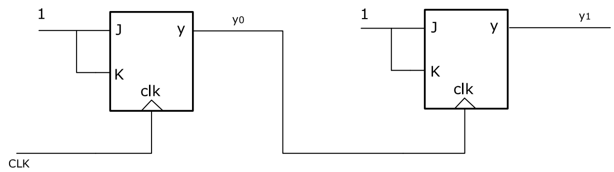


Figura A.117 (Esercizio A.35) Realizzazione del contatore con FFJK. L'uscita y_0 è un segnale periodico con frequenza pari a metà di quella del clock. Il segnale y_1 ha frequenza pari a metà di quella di y_0.

FFJK ciò si ottiene come in Figura A.117. Ovviamente se $n > 2$ si tratta di aggiungere il corrispondente numero di FF.

Nel caso di flip flop D, la divisione di frequenza del primo FF (y_0) comporta che esso deve cambiare stato ad ogni clock, ciò impone che l'ingresso sia il complemento dello stato. Dunque:

$$D_0 = \overline{y_0}$$

da cui deriva la rete di sinistra in Figura A.118. Per quanto riguarda il secondo FF, esso si comporterà come il precedente, ma userà come clock il segnale y_0. Ne deriva la rete a destra di Figura A.118.

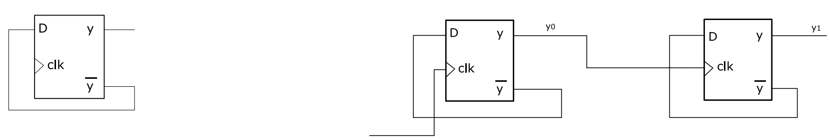


Figura A.118 (Esercizio A.35) A sinistra la divisione di frequenza in un FFD; a destra la realizzazione del contatore dell'Esercizio A.35 con FFD. L'uscita y_0 è un segnale periodico con frequenza pari a metà di quella del clock. Il segnale y_1 ha frequenza pari a metà di quella di y_0.

A.36 Per progettare un contatore sincrono modulo $N \neq 2^n$ conviene partire direttamente dalla tabella di flusso, si tratta di una tabella degenere, in quanto è priva di ingressi. In Figura A.119 vengono riportati la tabella di flusso di un generico contatore sincrono modulo N, quella di un contatore modulo 5 e la tabella delle transizioni di stato corrispondenti a quest'ultima (avendo codificato gli stati in modo naturale).

Dalla copertura delle mappe a sinistra in Figura A.120, ricaviamo le funzioni a destra nella stessa Figura. Mentre in Figura A.121 è mostrata la rete che realizza il contatore.

Sp	Sf
0	1
1	2
...	...
N-2	N-1
N-1	0

Sp	Sf
0	1
1	2
2	3
3	4
4	0

y_1y_0	$y'_1y'_0$
000	001
001	010
010	011
011	100
100	000

Figura A.119 (Esercizio A.36) Tabella di flusso di un contatore modulo N (a sinistra), tabella di flusso di un contatore modulo 5 (al centro) e transizione degli statio per quest'ultimo (a destra).

y'_0:

$y_0 y_1$ \ y_2	0	1
00	0	0
01	0	1
11	-	-
10	0	-

y'_1:

$y_0 y_1$ \ y_2	0	1
00	0	1
01	1	0
11	-	-
10	0	1

y'_2:

$y_0 y_1$ \ y_2	0	1
00	1	0
01	1	0
11	-	-
10	0	-

$$y'_0 = y_1 y_2$$
$$y'_1 = \overline{y_2} y_1 + \overline{y_1} y_2$$
$$y'_2 = \overline{y_0}\,\overline{y_2}$$

Figura A.120 (Esercizio A.36) Mappe e funzioni che realizzano il contatore modulo 5 .

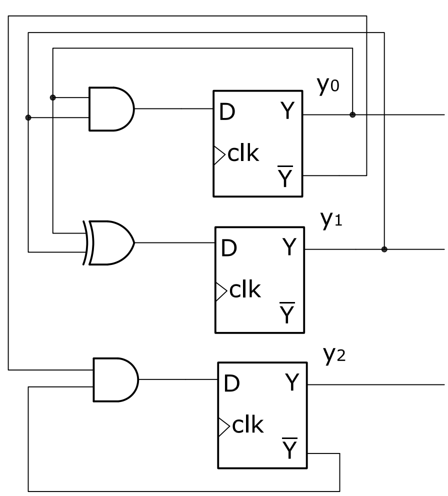


Figura A.121 (Esercizio A.36) Rete che realizza il contatore modulo 5.

Bibliografia

[Bar91] T. C. Bartee, *Computer architecture and logic design*, McGraw-Hill, 1991.

[BFRV92] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-programmable gate arrays*, Kluwer Academic publisher, New York, 1992.

[FSS07] F. Fummi, M. G. Sami, and C. Silvano, *Progettazione digitale*, Mc Graw-Hill, Milano, 2007, 2a edizione.

[KJ10] Z. Kohavi and N Jha, *Switching and finite automata theory*, Cambridge University Press, 2010, 4th edition.

[McC56] E. J. Jr. McCluskey, *Minimization of boolean functions*, Bell System Technical Journal **35** (1956), no. 5, 1417–1444.

[MK08] M. M. Mano and C. Kime, *Reti logiche*, Prentice Hall, 2008, 4 edizione.

[Won85] D. G. Wong, *Digital system design*, Edward Arnold, 1985.

Indice analitico

Indice